

DDDDDDDDDDDDDD		UUU		UUU	MMM		MMM	PPPPPPPPPPPP	
DDDDDDDDDDDDDD		UUU		UUU	MMM		MMM	PPPPPPPPPPPP	
DDDDDDDDDDDDDD		UUU		UUU	MMM		MMM	PPPPPPPPPPPP	
DDD	DDD	UUU		UUU	MMMMMM	MMMMMM	MMM	PPP	PPP
DDD	DDD	UUU		UUU	MMMMMM	MMMMMM	MMM	PPP	PPP
DDD	DDD	UUU		UUU	MMMMMM	MMMMMM	MMM	PPP	PPP
DDD	DDD	UUU		UUU	MMM	MMM	MMM	PPP	PPP
DDD	DDD	UUU		UUU	MMM	MMM	MMM	PPP	PPP
DDD	DDD	UUU		UUU	MMM	MMM	MMM	PPP	PPP
DDD	DDD	UUU		UUU	MMM	MMM	MMM	PPPPPPPPPPPP	
DDD	DDD	UUU		UUU	MMM	MMM	MMM	PPPPPPPPPPPP	
DDD	DDD	UUU		UUU	MMM	MMM	MMM	PPPPPPPPPPPP	
DDD	DDD	UUU		UUU	MMM	MMM	MMM	PPP	
DDD	DDD	UUU		UUU	MMM	MMM	MMM	PPP	
DDD	DDD	UUU		UUU	MMM	MMM	MMM	PPP	
DDD	DDD	UUU		UUU	MMM	MMM	MMM	PPP	
DDD	DDD	UUU		UUU	MMM	MMM	MMM	PPP	
DDD	DDD	UUU		UUU	MMM	MMM	MMM	PPP	
DDD	DDD	UUU		UUU	MMM	MMM	MMM	PPP	
DDD	DDD	UUU		UUU	MMM	MMM	MMM	PPP	
DDDDDDDDDDDDDD		UUUUUUUUUUUUUU		UUUU	MMM		MMM	PPP	
DDDDDDDDDDDDDD		UUUUUUUUUUUUUU		UUUU	MMM		MMM	PPP	
DDDDDDDDDDDDDD		UUUUUUUUUUUUUU		UUUU	MMM		MMM	PPP	

```
DDDDDDDD  UU  UU  MM  MM  P P P P P P P
DDDDDDDD  UU  UU  MM  MM  P P P P P P P
DD  DD  UU  UU  MMMM  MMMM  PP  PP
DD  DD  UU  UU  MMMM  MMMM  PP  PP
DD  DD  UU  UU  MM  MM  MM  PP  PP
DD  DD  UU  UU  MM  MM  MM  PP  PP
DD  DD  UU  UU  MM  MM  MM  P P P P P P P
DD  DD  UU  UU  MM  MM  MM  P P P P P P P
DD  DD  UU  UU  MM  MM  MM  PP
DD  DD  UU  UU  MM  MM  MM  PP
DD  DD  UU  UU  MM  MM  MM  PP
DDDDDDDD  UUUUUUUUU  MM  MM  PP
DDDDDDDD  UUUUUUUUU  MM  MM  PP
                                     ....
                                     ....
                                     ....
                                     ....
```

```
LL  IIIIII  SSSSSSSS
LL  IIIIII  SSSSSSSS
LL  II  SS
LL  II  SS
LL  II  SS
LL  II  SS
LL  II  SSSSSS
LL  II  SSSSSS
LL  II  SS
LL  II  SS
LL  II  SS
LL  II  SS
LLLLLLLLLL  IIIIII  SSSSSSSS
LLLLLLLLLL  IIIIII  SSSSSSSS
```

```
1 0001 0 MODULE DUMPSMAIN ( ! File dump program
2 0002 0 IDENT='V04-000',
3 0003 0 MAIN=dump$start,
4 0004 0 ADDRESSING MODE(EXTERNAL=GENERAL,
5 0005 0 NONEXTERNAL=LONG_RELATIVE)
6 0006 0 ) =
7 0007 1 BEGIN
8 0008 1
9 0009 1
10 0010 1 *****
11 0011 1 *
12 0012 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
13 0013 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
14 0014 1 * ALL RIGHTS RESERVED.
15 0015 1 *
16 0016 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
17 0017 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
18 0018 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
19 0019 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
20 0020 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
21 0021 1 * TRANSFERRED.
22 0022 1 *
23 0023 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
24 0024 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
25 0025 1 * CORPORATION.
26 0026 1 *
27 0027 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
28 0028 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
29 0029 1 *
30 0030 1 *****
31 0031 1
32 0032 1
33 0033 1
34 0034 1 ++
35 0035 1
36 0036 1 FACILITY: File dump utility
37 0037 1
38 0038 1 ABSTRACT:
39 0039 1 This module contains the command processing and driver routines.
40 0040 1
41 0041 1 ENVIRONMENT:
42 0042 1 VAX native, user mode.
43 0043 1
44 0044 1 AUTHOR: Benn Schreiber, Stephen Zalewski CREATION DATE: 22-Jun-1981
45 0045 1
46 0046 1 MODIFIED BY:
47 0047 1
48 0048 1 V03-013 SHZ0002 Stephen H. Zalewski 12-Apr-1984
49 0049 1 Move all conflicting qualifier checking to DUMP.CLD.
50 0050 1
51 0051 1 V03-012 BLS0258 Benn Schreiber 5-Jan-1984
52 0052 1 Clear fab$l_xab after opening the file.
53 0053 1
54 0054 1 V03-011 SHZ0001 Stephen H. Zalewski 28-Jun-1983
55 0055 1 Have DUMP open files shared.
56 0056 1
57 0057 1 V03-010 MLJ0095 Martin L. Jack, 17-Aug-1982 18:26
```


Remove references to CLISEND_PARSE.

V03-009 LMP0038 L. Mark Pilant, 30-Jun-1982 13:55
Correct a problem that generated the BADSTART error message
if the EOF block on a file was zero.

V03-008 LMP0034 L. Mark Pilant, 28-Jun-1982 9:36
Fix a bug introduced by LMP0030 that resulted in an access
violation when doing wildcard dumps.

V03-007 LMP0030 L. Mark Pilant, 15-Jun-1982 10:35
Allow dumping of logical blocks on a Files-11 mounted disk.

V03-006 MLJ0081 Martin L. Jack, 24-Feb-1982 17:35
Lengthen DUMP\$GQ_TIME to avoid overwriting EXIT_STATUS.

V03-005 MLJ0059 Martin L. Jack, 6-Nov-1981 14:13
Properly handle EFBK of 0.

V03-004 MLJ0056 Martin L. Jack, 18-Oct-1981 23:31
Special case reading from terminals to allow ^Z to function.

V03-003 MLJ0046 Martin L. Jack, 21-Sep-1981 18:27
Allow for device name change between \$PARSE and \$OPEN.

V03-002 MLJ0045 Martin L. Jack, 10-Sep-1981 15:27
Set SQ0 bit where appropriate. Allow record mode dump of
network device. Allow DUMP/HEADER on tape.

V03-001 MLJ0033 Martin L. Jack, 23-Aug-1981 9:48
Extensive rewriting to finish implementation.

```

92      0091 1 LIBRARY 'SYSS$LIBRARY:STARLET';
93      0092 1 LIBRARY 'SYSS$LIBRARY:TPAMAC';
94      0093 1 REQUIRE 'SRC$:DUMPRE';
95      0209 1
96      0210 1
97      0211 1 FORWARD ROUTINE
98      0212 1     dump$handler,      | Top-level condition handler
99      0213 1     dump$start,        | Main routine
100     0214 1     dump$tparse,       | Call TPARSE
101     0215 1     dump$store_num,    | Store numeric qualifier value
102     0216 1     dump$open_input,   | Open input file
103     0217 1     dump$open_output,  | Open output file
104     0218 1     dump$read,         | Read from input file
105     0219 1     dump$write:        NOVALUE, | Write to output file
106     0220 1     dump$close_input:   NOVALUE, | Close input file
107     0221 1     dump$close_output: NOVALUE, | Close output file
108     0222 1     dump$list_width:   NOVALUE, | Get listing device width
109     0223 1     dump$file_error:   NOVALUE; | Signal file-related error
110     0224 1
111     0225 1
112     0226 1 EXTERNAL ROUTINE
113     0227 1     cli$get_value,      | Get qualifier value
114     0228 1     cli$present,       | Test if qualifier present
115     0229 1     dump$blank_line,   | Write blank line
116     0230 1     dump$dump_file,    | Dump the file
117     0231 1     dump$output_getmsg, | Output a message
118     0232 1     lib$free_vm,       | Free virtual memory
119     0233 1     lib$get_vm,        | Allocate virtual memory
120     0234 1     lib$find_file,     | Search for wild card files
121     0235 1     lib$lp_lines,      | Number of lines on printer
122     0236 1     lib$tparse,       | Table-driven parser
123     0237 1     str$copy_dx;       | Copy a string
124     0238 1
125     0239 1
126     0240 1 EXTERNAL LITERAL
127     0241 1     dump$_facility,
128     0242 1     dump$_badrange,
129     0243 1     dump$_confqual,
130     0244 1     dump$_devquals,
131     0245 1     dump$_devspec,
132     0246 1     dump$_getchn,
133     0247 1     dump$_endoffile,
134     0248 1     dump$_novirmem,
135     0249 1     dump$_badstart;
136     0250 1
137     0251 1
138     0252 1 GLOBAL
139     0253 1     dump$gl_ifab : REF BBLOCK, | Pointer to input FAB
140     0254 1     dump$gl_inam : REF BBLOCK, | Pointer to input NAM block
141     0255 1     dump$gl_irab : $RAB_DECL,  | Input RAB
142     0256 1     dump$gl_orab : $RAB_DECL,  | Output RAB
143     0257 1     dump$gl_ofab : $FAB_DECL,  | Output FAB
144     0258 1     dump$gl_onam : $NAM_DECL,  | Output NAM block
145     0259 1     dump$gl_orss : BBLOCK[nam$c_maxrss], | Output resultant string
146     0260 1     dump$gl_idesc : BBLOCK[dsc$c_s_bln], | Descriptor for input RSA
147     0261 1     dump$gl_odesc : BBLOCK[dsc$c_s_bln], | Descriptor for output RSA
148     0262 1     dump$ab_outbuf : BBLOCK[dump$c_maxlisiz], | Output buffer
```

```
: 149      0263 1      dump$gl_outdesc : BBLOCK[dsc$c_s_bln],      : Descriptor for output buffer
: 150      0264 1      dump$gl_channel,      : Input channel
: 151      0265 1      dump$gl_width,      : Width of listing
: 152      0266 1      dump$gl_lpp,      : Lines per page
: 153      0267 1      dump$gl_buffer : BBLOCK[dsc$c_s_bln],      : Descriptor for input buffer
: 154      0268 1      dump$gl_flags : BBLOCK[4],      : General flags
: 155      0269 1      dump$gl_start_qual,      : Value of START qualifier
: 156      0270 1      dump$gl_end_qual,      : Value of END qualifier
: 157      0271 1      dump$gl_count_qual,      : Value of COUNT qualifier
: 158      0272 1      dump$gl_number_qual,      : Value of NUMBER qualifier
: 159      0273 1      dump$gl_number,      : Local byte offset for NUMBER
: 160      0274 1      dump$gl_cur_block,      : Current block number
: 161      0275 1      dump$gl_max_block,      : Highest block to be dumped
: 162      0276 1      dump$gl_file_efblk,      : End of file block
: 163      0277 1      dump$gl_file_hiblk,      : Highest allocated block
: 164      0278 1      dump$gl_record,      : Current block/record number
: 165      0279 1      dump$gl_time : VECTOR[2];      : Time at beginning of dump
: 166      0280 1
: 167      0281 1
: 168      0282 1      OWN
: 169      0283 1      exit_status : BBLOCK[4] INITIAL(ss$_normal),! Most severe error status
: 170      0284 1      tpa_block : BBLOCK[tpa$k_length0];      : TPARSE block
```



```

: 172      0285 1 LITERAL
: 173      0286 1
: 174      0287 1      dump$m_tpa_start=      $fieldmask(dump$v_tpa_start),
: 175      0288 1      dump$m_tpa_count=      $fieldmask(dump$v_tpa_count),
: 176      0289 1      dump$m_tpa_end=      $fieldmask(dump$v_tpa_end);
: 177      0290 1
: 178      0291 1 ! TPARSE tables to parse /BLOCK and /RECORD qualifier values.
: 179      0292 1 !
: 180      0293 1 $INIT STATE(blkrec_states, blkrec_keys);
: 181      P 0294 1 $STATE(
: 182      P 0295 1      ('START',,,dump$m_tpa_start,dump$gl_flags),
: 183      P 0296 1      ('END',,,dump$m_tpa_end, dump$gl_flags),
: 184      0297 1      ('COUNT',,,dump$m_tpa_count,dump$gl_flags));
: 185      P 0298 1 $STATE(
: 186      P 0299 1      ('='),
: 187      0300 1      (':'));
: 188      P 0301 1 $STATE(parse_number,
: 189      P 0302 1      (tpa$_decimal,eos,dump$store_num),      ! Decimal number
: 190      0303 1      ('X'));      ! Base prefix
: 191      P 0304 1 $STATE(
: 192      P 0305 1      ('X'),      ! Hex base designator
: 193      P 0306 1      ('O',octnum),      ! Octal base designator
: 194      0307 1      ('D',decnum));      ! Decimal base designator
: 195      P 0308 1 $STATE(
: 196      0309 1      (tpa$_hex,eos,dump$store_num));      ! Introduced hex number
: 197      P 0310 1 $STATE(octnum,
: 198      0311 1      (tpa$_octal,eos,dump$store_num));      ! Introduced octal number
: 199      P 0312 1 $STATE(decnum,
: 200      0313 1      (tpa$_decimal,,dump$store_num));      ! Introduced decimal number
: 201      P 0314 1 $STATE(eos,
: 202      0315 1      (tpa$_eos,tpa$_exit));      ! End of string
: 203      0316 1
: 204      0317 1
: 205      0318 1 ! TPARSE table to parse /NUMBER qualifier.
: 206      0319 1 !
: 207      0320 1 $INIT STATE(number_states, number_keys);
: 208      P 0321 1 $STATE(
: 209      0322 1      ((parse_number),tpa$_exit));      ! /NUMBER=
```

```
211 0323 1 ROUTINE dump$handler(sigargs, mechargs)=
212 0324 BEGIN
213 0325
214 0326 This routine is a condition handler established by the main
215 0327 routine. It saves the most severe condition for the exit status.
216 0328
217 0329 MAP
218 0330 sigargs : REF BBLOCK,
219 0331 mechargs : REF BBLOCK;
220 0332 BIND
221 0333 signame = sigargs[chf$l_sig_name] : BBLOCK; ! Name of signal
222 0334
223 0335
224 0336 IF NOT .signame ! If an error signal
225 0337 AND ((.signame[sts$v_severity] ! and severity is worse
226 0338 GTRU .exit_status[sts$v_severity])
227 0339 OR .exit_status[sts$v_severity]) ! or no errors yet
228 0340 THEN
229 0341 exit_status = .signame; ! then save it for exit
230 0342
231 0343
232 0344 RETURN ss$_resignal; ! Resignal to get message
233 0345 1 END; ! Of dump$handler
```

```
.TITLE DUMPSMAIN
.IDENT \V04-000\
```

```
.PSECT _LIB$KEY1$,NOWRT, SHR, PIC,1
```

```
00000 :TPASKEYSTO
U.2: .BLKB 0
54 52 41 54 53 00000 :TPASKEYST
U.4: .ASCII \START\
FF 00005 :TPASKEYST
00006 :TPASKEYSTO
U.8: .BLKB 0
44 4E 45 00006 :TPASKEYST
U.10: .ASCII \END\
FF 00009 :TPASKEYST
0000A :TPASKEYSTO
U.14: .BLKB 0
54 4E 55 4F 43 0000A :TPASKEYST
U.16: .ASCII \COUNT\
FF 0000F :TPASKEYST
FF 00010 :TPASKEYFILL
U.20: .BYTE -1
```

```
.PSECT _LIB$STATES,NOWRT, SHR, PIC,1
```

```
00000 BLKREC_STATES::
6100 00000 :TPASTYPE
U.5: .WORD 24832
00000000* 00002 :TPASADDR
U.6: .LONG <<DUMPSGL_FLAGS-U.6>-4>
10000000 00006 :TPASMASK
```


6101	0000A	U.7: LONG	268435456	:
		:TPASTYPE		:
00000000*	0000C	U.11: WORD	24833	:
		:TPASADDR		:
20000000	00010	U.12: LONG	<<DUMP\$GL_FLAGS-U.12>-4>	:
		:TPASMASK		:
6502	00014	U.13: LONG	536870912	:
		:TPASTYPE		:
00000000*	00016	U.17: WORD	25858	:
		:TPASADDR		:
40000000	0001A	U.18: LONG	<<DUMP\$GL_FLAGS-U.18>-4>	:
		:TPASMASK		:
003D	0001E	U.19: LONG	1073741824	:
		:TPASTYPE		:
043A	00020	U.21: WORD	61	:
		:TPASTYPE		:
	00022	U.22: WORD	1082	:
		PARSE_NUMBER:		:
		BLKB	0	:
91F3	00022	:TPASTYPE		:
		U.23: WORD	-28173	:
00000000V	00024	:TPASACTION		:
		U.24: LONG	<<DUMP\$STORE_NUM-U.24>-4>	:
0000*	00028	:TPASTARGET		:
		U.26: WORD	<<U.25-U.26>-2>	:
0425	0002A	:TPASTYPE		:
		U.27: WORD	1061	:
0058	0002C	:TPASTYPE		:
		U.28: WORD	88	:
104F	0002E	:TPASTYPE		:
		U.29: WORD	4175	:
0000*	00030	:TPASTARGET		:
		U.31: WORD	<<U.30-U.31>-2>	:
1444	00032	:TPASTYPE		:
		U.32: WORD	5188	:
0000*	00034	:TPASTARGET		:
		U.34: WORD	<<U.33-U.34>-2>	:
95F5	00036	:TPASTYPE		:
		U.35: WORD	-27147	:
00000000V	00038	:TPASACTION		:
		U.36: LONG	<<DUMP\$STORE_NUM-U.36>-4>	:
0000*	0003C	:TPASTARGET		:
		U.37: WORD	<<U.25-U.37>-2>	:
	0003E	OCTNUM		:
		U.30: BLKB	0	:
95F4	0003E	:TPASTYPE		:
		U.38: WORD	-27148	:
00000000V	00040	:TPASACTION		:
		U.39: LONG	<<DUMP\$STORE_NUM-U.39>-4>	:
0000*	00044	:TPASTARGET		:
		U.40: WORD	<<U.25-U.40>-2>	:
	00046	DECNUM		:
		U.33: BLKB	0	:
85F3	00046	:TPASTYPE		:
		U.41: WORD	-31245	:
00000000V	00048	:TPASACTION		:
		U.42: LONG	<<DUMP\$STORE_NUM-U.42>-4>	:

```
0004C :EOS
15F7 0004C U.25: .BLKB 0
          :TPASTYPE
          U.43: .WORD 5623
FFFF 0004E :TPASTARGET
          U.44: .WORD -1
          00050 NUMBER_STATES::
          : .BLKB 0
1DF8 00050 :TPASTYPE
          U.46: .WORD 7672
0000* 00052 :TPASSUBEXP
          U.47: .WORD <<PARSE_NUMBER-U.47>-2>
FFFF 00054 :TPASTARGET
          U.48: .WORD -1
          :
          .PSECT _LIB$KEY0$,NOWRT, SHR, PIC,1
          00000 BLKREC_KEYS::
          : .BLKB 0
          00000 :TPASKEY0
          U.1: .BLKB 0
0000* 00000 :TPASKEY
          U.3: .WORD <U.2-U.1>
0000* 00002 :TPASKEY
          U.9: .WORD <U.8-U.1>
0000* 00004 :TPASKEY
          U.15: .WORD <U.14-U.1>
          00006 : .BLKB 2
          00008 NUMBER_KEYS::
          : .BLKB 0
          00008 :TPASKEY0
          U.45: .BLKB 0
          :
          .PSECT $OWNS$,NOEXE,2
00000001 00000 EXIT_STATUS:
          : .LONG 1
          00004 TPA_BLOCK:
          : .BLKB 36
          :
          .PSECT $GLOBAL$,NOEXE,2
          00000 DUMP$GL_IFAB::
          : .BLKB 4
          00004 DUMP$GL_INAM::
          : .BLKB 4
          00008 DUMP$GL_IRAB::
          : .BLKB 68
          0004C DUMP$GL_ORAB::
          : .BLKB 68
          00090 DUMP$GL_OFAB::
          : .BLKB 80
          000E0 DUMP$GL_ONAM::
          : .BLKB 96
          00140 DUMP$GL_ORSS::
          : .BLKB 255
          0023F : .BLKB 1
```

00240 DUMPSGL_IDESC::
 .BKLB 8
00248 DUMPSGL_ODESC::
 .BKLB 8
00250 DUMPSAB_OUTBUF::
 .BKLB 132
002D4 DUMPSGL_OUTDESC::
 .BKLB 8
002DC DUMPSGL_CHANNEL::
 .BKLB 4
002E0 DUMPSGL_WIDTH::
 .BKLB 4
002E4 DUMPSGL_LPP::
 .BKLB 4
002E8 DUMPSGL_BUFFER::
 .BKLB 8
002F0 DUMPSGL_FLAGS::
 .BKLB 4
002F4 DUMPSGL_START_QUAL::
 .BKLB 4
002F8 DUMPSGL_END_QUAL::
 .BKLB 4
002FC DUMPSGL_COUNT_QUAL::
 .BKLB 4
00300 DUMPSGL_NUMBER_QUAL::
 .BKLB 4
00304 DUMPSGL_NUMBER::
 .BKLB 4
00308 DUMPSGL_CUR_BLOCK::
 .BKLB 4
0030C DUMPSGL_MAX_BLOCK::
 .BKLB 4
00310 DUMPSGL_FILE_EFBLK::
 .BKLB 4
00314 DUMPSGL_FILE_HIBLK::
 .BKLB 4
00318 DUMPSGL_RECORD::
 .BKLB 4
0031C DUMPSGL_TIME::
 .BKLB 8

.EXTRN CLISGET VALUE, CLISPRESENT
.EXTRN DUMPSBLANK LINE
.EXTRN DUMPSDUMP FILE, DUMPSOUTPUT_GETMSG
.EXTRN LIB\$FREE_VM, LIB\$GET_VM
.EXTRN LIB\$FIND_FILE, LIB\$LP_LINES
.EXTRN LIB\$PARSE, STR\$COPY_OX
.EXTRN DUMPS_FACILITY, DUMPS_BADRANGE
.EXTRN DUMPS_CONFQUAL, DUMPS_DEVQUALS
.EXTRN DUMPS_DEVSPEC, DUMPS_GETCHN
.EXTRN DUMPS_ENDOFFILE
.EXTRN DUMPS_NOVIRMEM, DUMPS_BADSTART
.PSECT \$CODE\$,NOWRT,2

0004 00000 DUMPSHANDLER:
 .WORD

Save R2

: 0323

DUMPSMAIN
V04-000

L 10
16-Sep-1984 01:26:41
14-Sep-1984 12:21:35

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[DUMP.SRC]DUMP.B32;1

Page 10
(4)

51	50	04	52 00000000'	EF	9E	00002	MOVAB	EXIT_STATUS, R2	:	0333
51	62		AC	04	C1	00009	ADDL3	#4, SIGARGS, R0	:	0336
	60		12	60	EB	0000E	BLBS	(R0), 2\$:	0338
			03	00	EF	00011	EXTZV	#0, #3, EXIT_STATUS, R1	:	
			03	00	ED	00016	CMPZV	#0, #3, (R0), R1	:	
				03	1A	00018	BGTRU	1\$:	
			03	62	E9	0001D	BLBC	EXIT_STATUS, 2\$:	0339
			62	60	D0	00020	MOVL	(R0), EXIT_STATUS	:	0341
			50	0918	8F	3C	MOVZWL	#2328, R0	:	0344
					04	00028	RET		:	0345

; Routine Size: 41 bytes, Routine Base: \$CODE\$ + 0000

```
235 0346 1 ROUTINE dump$start=
236 0347 BEGIN
237 0348
238 0349 This is the main program. It gathers all the command inputs, and then
239 0350 dumps the requested files.
240 0351
241 0352 LOCAL
242 0353 find_context : REF BBLOCK,           ! Context for lib$find file
243 0354 find_result : BBLOCK[dsc$c_s_bln], ! Result of lib$find file
244 0355 find_related : BBLOCK[dsc$c_s_bln], ! Related file for find_file
245 0356 find_default : BBLOCK[dsc$c_s_bln], ! Default file
246 0357 value_desc : BBLOCK[dsc$c_s_bln], ! Qualifier value descriptor
247 0358 status, ! Status variable
248 0359 output_desc : BBLOCK[dsc$c_s_bln], ! Output file specification
249 0360 input_desc : BBLOCK[dsc$c_s_bln], ! Input file specification
250 0361 input_devchar : BLOCK[16,BYTE]; ! $GETDVI arg block
251 0362
252 0363 BUILTIN
253 0364 fp;
254 0365
255 0366 .fp = dump$handler; ! Enable the condition handler
256 0367
257 0368
258 0369 CH$FILL(0, dsc$c_s_bln, input_desc);
259 0370 input_desc[dsc$b_class] = dsc$k_class_d; ! Make descriptors dynamic
260 0371 CH$MOVE(dsc$c_s_bln, input_desc, output_desc);
261 0372 CH$MOVE(dsc$c_s_bln, input_desc, find_result);
262 0373 CH$MOVE(dsc$c_s_bln, input_desc, find_related);
263 0374 CH$MOVE(dsc$c_s_bln, input_desc, find_default);
264 0375 CH$MOVE(dsc$c_s_bln, input_desc, value_desc);
265 0376
266 0377
267 0378 ! Get parameters and qualifiers.
268 0379
269 0380 cli$get_value($descriptor('INPUT'), input_desc); ! Get input file spec
270 0381 cli$get_value($descriptor('OUTPUT'), output_desc); ! Get output file spec
271 0382 dump$gl_flags[dump$sv_allocated] = cli$present($descriptor('ALLOCATED'));
272 0383 dump$gl_flags[dump$sv_blocks] = cli$present($descriptor('BLOCKS'));
273 0384 dump$gl_flags[dump$sv_byte] = cli$present($descriptor('BYTE'));
274 0385 dump$gl_flags[dump$sv_decimal] = cli$present($descriptor('DECIMAL'));
275 0386 dump$gl_flags[dump$sv_file_header] = cli$present($descriptor('FILE HEADER'));
276 0387 dump$gl_flags[dump$sv_formatted] = cli$present($descriptor('FORMATTED'));
277 0388 dump$gl_flags[dump$sv_header] = cli$present($descriptor('HEADER'));
278 0389 dump$gl_flags[dump$sv_hex] = cli$present($descriptor('HEXADECIMAL'));
279 0390 dump$gl_flags[dump$sv_longword] = cli$present($descriptor('LONGWORD'));
280 0391 dump$gl_flags[dump$sv_number] = cli$present($descriptor('NUMBER'));
281 0392 dump$gl_flags[dump$sv_octal] = cli$present($descriptor('OCTAL'));
282 0393 dump$gl_flags[dump$sv_output] = cli$present($descriptor('OUTPUT'));
283 0394 dump$gl_flags[dump$sv_printer] = cli$present($descriptor('PRINTER'));
284 0395 dump$gl_flags[dump$sv_records] = cli$present($descriptor('RECORDS'));
285 0396 dump$gl_flags[dump$sv_word] = cli$present($descriptor('WORD'));
286 0397
287 0398
288 0399
289 0400 ! If /NUMBER qualifier is present, get the value.
290 0401
291 0402 IF .dump$gl_flags[dump$sv_number] ! /NUMBER present
```

```
292 0403 THEN
293 0404 BEGIN
294 0405 IF cli$get_value($descriptor('NUMBER'), value_desc)
295 0406 THEN
296 0407 BEGIN
297 0408 dump$gl_flags[dump$vp_tpa_number] = true; ! Note parsing /NUMBER
298 0409 IF NOT dump$tparse(value_desc, number_states, number_keys)
299 0410 THEN
300 0411 SIGNAL_STOP(
301 0412 dump$facility*16 + shr$syntax + sts$severe,
302 0413 1, value_desc);
303 0414 END;
304 0415 END;
305 0416
306 0417
307 0418 ! If /BLOCK qualifier is present, get the value(s).
308 0419
309 0420 IF .dump$gl_flags[dump$vp_blocks] ! /BLOCKS present
310 0421 THEN
311 0422 BEGIN
312 0423 WHILE cli$get_value($descriptor('BLOCKS'), value_desc) DO
313 0424 BEGIN
314 0425 IF NOT dump$tparse(value_desc, blkrec_states, blkrec_keys)
315 0426 THEN
316 0427 SIGNAL_STOP(
317 0428 dump$facility*16 + shr$syntax + sts$severe,
318 0429 1, value_desc);
319 0430 END;
320 0431 END;
321 0432
322 0433
323 0434 ! If /RECORD qualifier is present, get the value(s).
324 0435
325 0436 IF .dump$gl_flags[dump$vp_records] ! /RECORDS present
326 0437 THEN
327 0438 BEGIN
328 0439 WHILE cli$get_value($descriptor('RECORDS'), value_desc) DO
329 0440 BEGIN
330 0441 IF NOT dump$tparse(value_desc, blkrec_states, blkrec_keys)
331 0442 THEN
332 0443 SIGNAL_STOP(
333 0444 dump$facility*16 + shr$syntax + sts$severe,
334 0445 1, value_desc);
335 0446 END;
336 0447 END;
337 0448
338 0449
339 0450 ! Check range of START and END if both were specified, to ensure that START
340 0451 is less than END.
341 0452
342 0453 IF .dump$gl_flags[dump$vp_start] AND .dump$gl_flags[dump$vp_end]
343 0454 AND .dump$gl_start_qual GTRU .dump$gl_end_qual
344 0455 THEN
345 0456 SIGNAL_STOP(dump$badrange);
346 0457
347 0458
348 0459 ! Get number of lines on output page.
```



```
0460 !
0461 dump$gl_lpp = lib$lpp_lines() - 6;
0462
0463
0464 ! Loop, calling LIB$FIND_FILE to get files matching the input spec.
0465
0466 find_context = 0; ! Initialize context
0467 UNTIL
0468 BEGIN
0469 status = lib$find_file(
0470 input_desc,
0471 find_result,
0472 find_context,
0473 find_default,
0474 find_related);
0475 IF .find_context NEQ 0 THEN dump$gl_inam = .find_context[fab$l_inam];
0476 IF .status EQL rms$_dnf OR .status EQL rms$_fnf
0477 THEN
0478 BEGIN
0479 IF (.dump$gl_inam[nam$l_fnb] AND ! Check for only device
0480 (nam$m_exp_dir OR
0481 nam$m_exp_name OR
0482 nam$m_exp_type OR
0483 nam$m_exp_ver OR
0484 nam$m_wildcard)) EQL 0
0485 THEN
0486 BEGIN
0487 input_devchar[dumpdvi_w_size] = 4; ! Build $GETDVI item
0488 input_devchar[dumpdvi_w_type] = dvi$_devchar; ! list for the
0489 input_devchar[dumpdvi_l_addr] = find_context[fab$l_dev]; ! device
0490 input_devchar[dumpdvi_l_len] = 0; ! characteristics
0491 input_devchar[dumpdvi_l_end] = 0; ! Terminate the list
0492 $GETDVI(EFN = dumpdvi_c_efn, ! Get characteristics
0493 DEVNAM = input_desc,
0494 ITMLST = input_devchar);
0495 $WAITFR(EFN = dumpdvi_c_efn); ! Wait until complete
0496 status = 1; ! Don't take an error
0497 BBLOCK[find_context[fab$l_dev], dev$v_for] = 1; ! Mark foreign
0498 END;
0499 END;
0500 .status EQL rms$_nmf
0501 END
0502 DO
0503 BEGIN
0504 IF NOT .status ! Report error
0505 THEN
0506 BEGIN
0507 SIGNAL(
0508 dump$facility*16 + shr$_openin + sts$_error,
0509 1, find_result,
0510 .find_context[fab$l_sts], .find_context[fab$l_stv]);
0511 END
0512 ELSE
0513 BEGIN
0514 IF dump$open_input(.find_context, find_result)
0515 AND dump$open_output(output_desc, .find_context)
0516 THEN
```

```

406      BEGIN
407      dump$list_width(dump$gl_ofab);
408      $GETTIM(t$tmadr=dump$gl_time);
409      dump$gl_number = .dump$gl_number_qual;
410      dump$dump_file();
411      dump$close_input(.find_context);
412      dump$close_output();
413      END;
414      str$copy_dx(find_related, find_result);
415      END;
416      IF NOT .dump$gl_inam[nam$w wildcard]
417      THEN RETURN .exit_status OR sts$m_inhib_msg;
418      END;
419      RETURN .exit_status OR sts$m_inhib_msg;
420      END;
421
422
```

.PSECT SPLITS, NOWRT, NOEXE, 2

```

54 55 50 4E 49 00000 P.AAB: .ASCII \INPUT\
00005 .BLKB 3
00000005 00008 P.AAA: .LONG 5
00000000 0000C .ADDRESS P.AAB
54 55 50 54 55 4F 00010 P.AAD: .ASCII \OUTPUT\
00016 .BLKB 2
00000006 00018 P.AAC: .LONG 6
00000000 0001C .ADDRESS P.AAD
44 45 54 41 43 4F 4C 4C 41 00020 P.AAF: .ASCII \ALLOCATED\
00029 .BLKB 3
00000009 0002C P.AAE: .LONG 9
00000000 00030 .ADDRESS P.AAF
49 49 43 53 41 00034 P.AAH: .ASCII \ASCII\
00039 .BLKB 3
00000005 0003C P.AAG: .LONG 5
00000000 00040 .ADDRESS P.AAH
53 4B 43 4F 4C 42 00044 P.AAJ: .ASCII \BLOCKS\
0004A .BLKB 2
00000006 0004C P.AAI: .LONG 6
00000000 00050 .ADDRESS P.AAJ
45 54 59 42 00054 P.AAL: .ASCII \BYTE\
00000004 00058 P.AAK: .LONG 4
00000000 0005C .ADDRESS P.AAL
4C 41 4D 49 43 45 44 00060 P.AAN: .ASCII \DECIMAL\
00067 .BLKB 1
00000007 00068 P.AAM: .LONG 7
00000000 0006C .ADDRESS P.AAN
52 45 44 41 45 4B 5F 45 4C 49 46 00070 P.AAP: .ASCII \FILE_HEADER\
0007B .BLKB 1
00000008 0007C P.AAO: .LONG 11
00000000 00080 .ADDRESS P.AAP
44 45 54 54 41 4D 52 4F 46 00084 P.AAR: .ASCII \FORMATTED\
0008D .BLKB 3
00000009 00090 P.AAQ: .LONG 9
00000000 00094 .ADDRESS P.AAR
```

```
52 45 44 41 45 48 00098 P.AAT: .ASCII \HEADER\
0009E .BLKB 2
000A0 P.AAS: .LONG 6
000A4 .ADDRESS P.AAT
4C 41 4D 49 43 45 44 41 58 45 48 000A8 P.AAV: .ASCII \HEXADECIMAL\
000B3 .BLKB 1
0000000B 000B4 P.AAU: .LONG 11
00000000 000B8 .ADDRESS P.AAV
44 52 4F 57 47 4E 4F 4C 000BC P.AAX: .ASCII \LONGWORD\
00000008 0C0C4 P.AAW: .LONG 8
00000000 000C8 .ADDRESS P.AAX
52 45 42 4D 55 4E 000CC P.AAZ: .ASCII \NUMBER\
000D2 .BLKB 2
00000006 000D4 P.AAY: .LONG 6
00000000 000D8 .ADDRESS P.AAZ
4C 41 54 43 4F 000DC P.ABB: .ASCII \OCTAL\
000E1 .BLKB 3
00000005 000E4 P.ABA: .LONG 5
00000000 000E8 .ADDRESS P.ABB
54 55 50 54 55 4F 000EC P.ABD: .ASCII \OUTPUT\
000F2 .BLKB 2
00000006 000F4 P.ABC: .LONG 6
00000000 000F8 .ADDRESS P.ABD
52 45 54 4E 49 52 50 000FC P.ABF: .ASCII \PRINTER\
00103 .BLKB 1
00000007 00104 P.ABE: .LONG 7
00000000 00108 .ADDRESS P.ABF
53 44 52 4F 43 45 52 0010C P.ABH: .ASCII \RECORDS\
00113 .BLKB 1
00000007 00114 P.ABG: .LONG 7
00000000 00118 .ADDRESS P.ABH
44 52 4F 57 0011C P.ABJ: .ASCII \WORD\
00000004 00120 P.ABI: .LONG 4
00000000 00124 .ADDRESS P.ABJ
52 45 42 4D 55 4E 00128 P.ABL: .ASCII \NUMBER\
0012E .BLKB 2
00000006 00130 P.ABK: .LONG 6
00000000 00134 .ADDRESS P.ABL
53 4B 43 4F 4C 42 00138 P.ABN: .ASCII \BLOCKS\
0013E .BLKB 2
00000006 00140 P.ABM: .LONG 6
00000000 00144 .ADDRESS P.ABN
53 44 52 4F 43 45 52 00148 P.ABP: .ASCII \RECORDS\
0014F .BLKB 1
00000007 00150 P.ABO: .LONG 7
00000000 00154 .ADDRESS P.ABP
```

.EXTRN SYS\$GETDVI, SYS\$WAITFR

.EXTRN SYS\$GETTIM

.PSECT \$CODE\$,NOWRT,2

OFFC 0000 DUMP\$START:

```
5B 00000000 EF 9E 00002
5A 00000000 00 9E 00009
59 00000000 00 9E 00010
```

```
.WORD Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11
MOVAB BLKREC KEYS, R11
MOVAB LIB$STOP, R10
MOVAB CLIS$GET_VALUE, R9
```

0346

08	00	17	AE	58	00000000G	00	9E	00017	MOVAB	CLISPRESENT, R8	0366
		14	AE	57	000000000	EF	9E	0001E	MOVAB	P.AAA, R7	0369
				56	000000000	EF	9E	00025	MOVAB	DUMPSGL_FLAGS, R6	
				5E	BC	AE	9E	0002C	MOVAB	-68(SP), SP	
				6D	A4	AF	9E	00030	MOVAB	DUMPSHANDLER, (FP)	
				6E	14	00	2C	00034	MOVCS	#0, (SP), #0, #8, INPUT_DESC	
						AE		00039			
		17	AE			02	90	0003B	MOVB	#2, INPUT_DESC+3	0370
	1C	14	AE			08	28	0003F	MOVCS	#8, INPUT_DESC, OUTPUT_DESC	0371
	3C	14	AE			08	28	00045	MOVCS	#8, INPUT_DESC, FIND_RESULT	0372
	34	14	AE			08	28	0004B	MOVCS	#8, INPUT_DESC, FIND-RELATED	0373
	2C	14	AE			08	28	00051	MOVCS	#8, INPUT_DESC, FIND-DEFAULT	0374
	24	14	AE			08	28	00057	MOVCS	#8, INPUT_DESC, VALUE_DESC	0375
					14	AE	9F	0005D	PUSHAB	INPUT_DESC	0380
				69		57	DD	00060	PUSHL	R7	
					1C	02	FB	00062	CALLS	#2, CLISGET_VALUE	
					10	AE	9F	00065	PUSHAB	OUTPUT_DESC	0381
				69		A7	9F	00068	PUSHAB	P.AAC	
					24	02	FB	0006B	CALLS	#2, CLISGET_VALUE	0382
				68		A7	9F	0006E	PUSHAB	P.AAE	
66		01		00		01	FB	00071	CALLS	#1, CLISPRESENT	
					34	50	FD	00074	INSV	R0, #0, #1, DUMPSGL_FLAGS	0383
				68		A7	9F	00079	PUSHAB	P.AAG	
					44	01	FB	0007C	CALLS	#1, CLISPRESENT	0384
				68		A7	9F	0007F	PUSHAB	P.AAI	
66		01		01		01	FB	00082	CALLS	#1, CLISPRESENT	
					50	50	FD	00085	INSV	R0, #1, #1, DUMPSGL_FLAGS	0385
				68		A7	9F	0008A	PUSHAB	P.AAK	
66		01		02		01	FB	0008D	CALLS	#1, CLISPRESENT	
					60	50	FD	00090	INSV	R0, #2, #1, DUMPSGL_FLAGS	0386
				68		A7	9F	00095	PUSHAB	P.AAM	
66		01		03		01	FB	00098	CALLS	#1, CLISPRESENT	
					74	50	FD	0009B	INSV	R0, #3, #1, DUMPSGL_FLAGS	0387
				68		A7	9F	000A0	PUSHAB	P.AAO	
66		01		04		01	FB	000A3	CALLS	#1, CLISPRESENT	
					0088	50	FD	000A6	INSV	R0, #4, #1, DUMPSGL_FLAGS	0388
				68		C7	9F	000AB	PUSHAB	P.AAQ	
66		01		05		01	FB	000AF	CALLS	#1, CLISPRESENT	
					0098	50	FD	000B2	INSV	R0, #5, #1, DUMPSGL_FLAGS	0389
				68		C7	9F	000B7	PUSHAB	P.AAS	
66		01		06		01	FB	000BB	CALLS	#1, CLISPRESENT	
					00AC	50	FD	000BE	INSV	R0, #6, #1, DUMPSGL_FLAGS	0390
				68		C7	9F	000C3	PUSHAB	P.AAU	
66		01		07		01	FB	000C7	CALLS	#1, CLISPRESENT	
					00BC	50	FD	000CA	INSV	R0, #7, #1, DUMPSGL_FLAGS	0391
				68		C7	9F	000CF	PUSHAB	P.AAW	
01	A6	01		00		01	FB	000D3	CALLS	#1, CLISPRESENT	
					00CC	50	FD	000D6	INSV	R0, #0, #1, DUMPSGL_FLAGS+1	0392
				68		C7	9F	000DC	PUSHAB	P.AAY	
01	A6	01		01		01	FB	000E0	CALLS	#1, CLISPRESENT	
					00DC	50	FD	000E3	INSV	R0, #1, #1, DUMPSGL_FLAGS+1	0393
				68		C7	9F	000E9	PUSHAB	P.ABA	
01	A6	01		02		01	FB	000ED	CALLS	#1, CLISPRESENT	
					00EC	50	FD	000F0	INSV	R0, #2, #1, DUMPSGL_FLAGS+1	0394
				68		C7	9F	000F6	PUSHAB	P.ABC	
01	A6	01		03		01	FB	000FA	CALLS	#1, CLISPRESENT	
						50	FD	000FD	INSV	R0, #3, #1, DUMPSGL_FLAGS+1	

01	A6	01	68	00FC	C7	9F	00103	PUSHAB	P.ABE	0395
			04		01	FB	00107	CALLS	#1, CLISPRESNT	
				010C	50	FO	0010A	INSV	R0, #4, #1, DUMPSGL_FLAGS+1	0396
01	A6	01	68		C7	9F	00110	PUSHAB	P.ABG	
			05		01	FB	00114	CALLS	#1, CLISPRESNT	
				0118	50	FO	00117	INSV	R0, #5, #1, DUMPSGL_FLAGS+1	0397
01	A6	01	68		C7	9F	0011D	PUSHAB	P.ABI	
			06		01	FB	00121	CALLS	#1, CLISPRESNT	
		36	01	A6	50	FO	00124	INSV	R0, #6, #1, DUMPSGL_FLAGS+1	0402
				24	01	E1	0012A	BBC	#1, DUMPSGL_FLAGS+1, 1\$	0405
				0128	AE	9F	0012F	PUSHAB	VALUE_DESC	
			69		C7	9F	00132	PUSHAB	P.ABK	
			29		02	FB	00136	CALLS	#2, CLISGET_VALUE	
		03	A6	80	50	E9	00139	BLBC	R0, 1\$	0408
				08	8F	88	0013C	BISB2	#128, DUMPSGL_FLAGS+3	0409
				00000000*	AB	9F	00141	PUSHAB	NUMBER_KEYS	
				2C	EF	9F	00144	PUSHAB	NUMBER_STATES	
			00000000V	EF	AE	9F	0014A	PUSHAB	VALUE_DESC	
				0E	03	FB	0014D	CALLS	#3, DUMPSTPARSE	
				24	50	E8	00154	BLBS	R0, 1\$	0411
					AE	9F	00157	PUSHAB	VALUE_DESC	
				00000000*	01	DD	0015A	PUSHL	#1	0412
			6A		8F	DD	0015C	PUSHL	#<<<DUMPS FACILITY@16>+4344>+4>	
32			66		03	FB	00162	CALLS	#3, LIB\$STOP	0420
				24	01	E1	00165	BBC	#1, DUMPSGL_FLAGS, 3\$	0423
				0138	AE	9F	00169	PUSHAB	VALUE_DESC	
			69		C7	9F	0016C	PUSHAB	P.ABM	
			25		02	FB	00170	CALLS	#2, CLISGET_VALUE	
				00000000*	50	E9	00173	BLBC	R0, 3\$	0425
				2C	5B	DD	00176	PUSHL	R11	
			00000000V	EF	EF	9F	00178	PUSHAB	BLKREC_STATES	
				DE	AE	9F	0017E	PUSHAB	VALUE_DESC	
				24	03	FB	00181	CALLS	#3, DUMPSTPARSE	
					50	E8	00188	BLBS	R0, 2\$	0427
				00000000*	AE	9F	0018B	PUSHAB	VALUE_DESC	
			6A		01	DD	0018E	PUSHL	#1	0428
					8F	DD	00190	PUSHL	#<<<DUMPS FACILITY@16>+4344>+4>	
32					03	FB	00196	CALLS	#3, LIB\$STOP	0423
			01	A6	CE	11	00199	BRB	2\$	0436
				24	05	E1	0019B	BBC	#5, DUMPSGL_FLAGS+1, 5\$	0439
				0148	AE	9F	001A0	PUSHAB	VALUE_DESC	
			69		C7	9F	001A3	PUSHAB	P.ABO	
			25		02	FB	001A7	CALLS	#2, CLISGET_VALUE	
				00000000*	50	E9	001AA	BLBC	R0, 5\$	0441
				2C	5B	DD	001AD	PUSHL	R11	
			00000000V	EF	EF	9F	001AF	PUSHAB	BLKREC_STATES	
				DE	AE	9F	001B5	PUSHAB	VALUE_DESC	
				24	03	FB	001B8	CALLS	#3, DUMPSTPARSE	0443
					50	E8	001BF	BLBS	R0, 4\$	
				00000000*	AE	9F	001C2	PUSHAB	VALUE_DESC	0444
			6A		01	DD	001C5	PUSHL	#1	
					8F	DD	001C7	PUSHL	#<<<DUMPS FACILITY@16>+4344>+4>	0439
				01	03	FB	001CD	CALLS	#3, LIB\$STOP	0453
					CE	11	001D0	BRB	4\$	
				10	A6	95	001D2	TSTB	DUMPSGL_FLAGS+1	
					14	18	001D5	BGEQ	6\$	
					A6	E9	001D7	BLBC	DUMPSGL_FLAGS+2, 6\$	

08	A6	04	A6	D1	001DB	CMPL	DUMPSGL_START_QUAL, DUMPSGL_END_QUAL	0454
			09	18	001E0	BLEQU	6\$	
		00000000G	8F	DD	001E2	PUSHL	#DUMPS_BADRANGE	0456
	6A		01	FB	001E8	CALLS	#1, LIB\$STOP	
00000000G	00		00	FB	001EB	CALLS	#0, LIB\$LP_LINES	0461
F4	A6	FA	A0	9E	001F2	MOVAB	-6(R0), DUMPSGL_LPP	
			6E	D4	001F7	CLRL	FIND_CONTEXT	0466
		34	AE	9F	001F9	PUSHAB	FIND_RELATED	0469
		30	AE	9F	001FC	PUSHAB	FIND_DEFAULT	
		08	AE	9F	001FF	PUSHAB	FIND_CONTEXT	
		48	AE	9F	00202	PUSHAB	FIND_RESULT	
		24	AE	9F	00205	PUSHAB	INPUT_DESC	
00000000G	00		05	FB	00208	CALLS	#5, LIB\$FIND_FILE	
	53		50	DD	0020F	MOVL	R0, STATUS	
	52		6E	DD	00212	MOVL	FIND_CONTEXT, R2	0475
			06	13	00215	BEQL	8\$	
FD14	C6	28	A2	DD	00217	MOVL	40(R2), DUMPSGL_INAM	
0001C04A	8F		53	D1	0021D	CMPL	STATUS, #114762	0476
			09	13	00224	BEQL	9\$	
00018292	8F		53	D1	00226	CMPL	STATUS, #98962	
			41	12	0022D	BNEQ	10\$	
	50	FD14	C6	DD	0022F	MOVL	DUMPSGL_INAM, R0	0479
0147	8F	34	A0	B3	00234	BITW	52(R0), #327	0484
			34	12	0023A	BNEQ	10\$	
	AE	00020004	8F	DD	0023C	MOVL	#131076, INPUT_DEVCHAR	0487
04	AE	40	A2	9E	00244	MOVAB	64(R2), INPUT_DEVCHAR+4	0489
08		0C	AE	7C	00249	CLRQ	INPUT_DEVCHAR+8	0490
			7E	7C	0024C	CLRQ	-(SP)	0494
			7E	7C	0024E	CLRQ	-(SP)	
		14	AE	9F	00250	PUSHAB	INPUT_DEVCHAR	
		28	AE	9F	00253	PUSHAB	INPUT_DESC	
	7E		03	7D	00256	MOVQ	#3, -(SP)	
00000000G	00		08	FB	00259	CALLS	#8, SYS\$GETDVI	
			03	DD	00260	PUSHL	#3	0495
00000000G	00		01	FB	00262	CALLS	#1, SYS\$WAITFR	
	53		01	DD	00269	MOVL	#1, STATUS	0496
43	A2		01	88	0026C	BISB2	#1, 67(R2)	0497
000182CA	8F		53	D1	00270	CMPL	STATUS, #99018	0500
			03	12	00277	BNEQ	11\$	
		0083	31	00279	BRW	15\$		
			53	E8	0027C	BLBS	STATUS, 12\$	0504
	18	08	A2	7D	0027F	MOVQ	8(R2), -(SP)	0510
	7E	44	AE	9F	00283	PUSHAB	FIND_RESULT	0507
			01	DD	00286	PUSHL	#1	
		00000000*	8F	DD	00288	PUSHL	#<<<DUMPS_FACILITY@16>+4248>+2>	0508
00000000G	00		05	FB	0028E	CALLS	#5, LIB\$SIGNAL	
			5C	11	00295	BRB	14\$	0504
		3C	AE	9F	00297	PUSHAB	FIND_RESULT	0514
			52	DD	0029A	PUSHL	R2	
00000000V	EF		02	FB	0029C	CALLS	#2, DUMPSOPEN_INPUT	
	40		50	E9	002A3	BLBC	R0, 13\$	
			52	DD	002A6	PUSHL	R2	0515
		20	AE	9F	002AB	PUSHAB	OUTPUT_DESC	
00000000V	EF		02	FB	002AB	CALLS	#2, DUMPSOPEN_OUTPUT	
	31		50	E9	002B2	BLBC	R0, 13\$	
		FDA0	C6	9F	002B5	PUSHAB	DUMPSGL_OFAB	0518
00000000V	EF		01	FB	002B9	CALLS	#1, DUMPSLIST_WIDTH	

DUMPSMAIN
V04-000

H 11
16-Sep-1984 01:26:41
14-Sep-1984 12:21:35

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[DUMP.SRC]DUMP.B32;1

Page 19
(5)

00000000G	00	2C	A6	9F	002C0	PUSHAB	DUMPSGQ TIME	0519
14	A6	10	01	FB	002C3	CALLS	#1, SYS\$GETTIM	
00000000G	00		A6	DD	002CA	MOVL	DUMPSGL_NUMBER_QUAL, DUMPSGL_NUMBER	0520
			00	FB	002CF	CALLS	#0, DUMPSDUMP_FILE	0521
00000000V	EF		52	DD	002D6	PUSHL	R2	0522
00000000V	EF		01	FB	002D8	CALLS	#1, DUMPSCLOSE_INPUT	
			00	FB	002DF	CALLS	#0, DUMPSCLOSE_OUTPUT	0523
		3C	AE	9F	002E6	PUSHAB	FIND_RESULT	0525
		38	AE	9F	002E9	PUSHAB	FIND-RELATED	
00000000G	00		02	FB	002EC	CALLS	#2, STR\$COPY_DX	
	50	FD14	C6	DD	002F3	MOVL	DUMPSGL_INAM, R0	0527
	03	35	A0	E9	002F8	BLBC	53(R0), -158	
			FEFA	31	002FC	BRW	78	
50 00000000'	EF	10000000	8F	C9	002FF	BISL3	#268435456, EXIT_STATUS, R0	0532
			04	0030B		RET		0533

; Routine Size: 780 bytes, Routine Base: \$CODE\$ + 0029

```
0534 1 ROUTINE dump$tparse(string, states, keys)=
0535 BEGIN
0536
0537 This routine calls TPARSE given the string, states and keys.
0538
0539 Inputs:
0540
0541 string address of descriptor for string
0542 states address of TPARSE states table
0543 keys address of TPARSE keys table
0544
0545 MAP
0546 string : REF BBLOCK;
0547
0548
0549 CH$FILL(0, tpa$length0, tpa_block); ! Initialize block
0550 tpa_block[tpa$l_count] = tpa$length0;
0551 tpa_block[tpa$l_options] = tpa$m_abbrev;
0552 tpa_block[tpa$l_stringcnt] = .string[dsc$w_length];
0553 tpa_block[tpa$l_stringptr] = .string[dsc$a_pointer];
0554 RETURN lib$tparse(tpa_block, .states, .keys); ! Call TPARSE
0555 1 END;
```

007C 00000 DUMP\$TPARSE:									
									0534
		56	00000000	EF	9E	00002	WORD	Save R2,R3,R4,R5,R6	
24	00	6E		00	2C	00009	MOVAB	TPA_BLOCK, R6	
				66		0000E	MOVCS	#0, -(SP), #0, #36, TPA_BLOCK	0549
		66		08	D0	0000F	MOVL	#8, TPA_BLOCK	0550
	04	A6		02	D0	00012	MOVL	#2, TPA_BLOCK+4	0551
		50	04	AC	D0	00016	MOVL	STRING, R0	0552
	08	A6		60	3C	0001A	MOVZWL	(R0), TPA_BLOCK+8	
	0C	A6	04	A0	D0	0001E	MOVL	4(R0), TPA_BLOCK+12	0553
		7E	08	AC	7D	00023	MOVQ	STATES, -(SP)	0554
				56	DD	00027	PUSHL	R6	
		00000000G	00	03	FB	00029	CALLS	#3, LIB\$TPARSE	
				04	00	00030	RET		0555

; Routine Size: 49 bytes, Routine Base: \$CODE\$ + 0335

```

447 0556 1 ROUTINE dump$store_num=
448 0557 BEGIN
449 0558
450 0559 This routine is called when the /BLOCKS, /RECORDS, or /NUMBER qualifier
451 0560 is used. It interrogates flags set by TPARSE to determine which numeric
452 0561 value has been parsed and stores it in the appropriate result cell.
453 0562
454 0563 IF .dump$gl_flags[dump$v_tpa_start] ! Parsing START
455 0564 THEN
456 0565 BEGIN
457 0566 dump$gl_start_qual = .tpa_block[tpa$l_number];
458 0567 dump$gl_flags[dump$v_start] = true; ! Note START was present
459 0568 END
460 0569 ELSE IF .dump$gl_flags[dump$v_tpa_end] ! Parsing END
461 0570 THEN
462 0571 BEGIN
463 0572 dump$gl_end_qual = .tpa_block[tpa$l_number];
464 0573 dump$gl_flags[dump$v_end] = true; ! Note END was present
465 0574 END
466 0575 ELSE IF .dump$gl_flags[dump$v_tpa_count] ! Parsing COUNT
467 0576 THEN
468 0577 BEGIN
469 0578 dump$gl_count_qual = .tpa_block[tpa$l_number];
470 0579 dump$gl_flags[dump$v_count] = true; ! Note COUNT was present
471 0580 END
472 0581 ELSE IF .dump$gl_flags[dump$v_tpa_number] ! Parsing NUMBER
473 0582 THEN
474 0583 dump$gl_number_qual = .tpa_block[tpa$l_number]
475 0584 ELSE
476 0585 SIGNAL_STOP(dump$_facility*16 + shr$_badlogic + sts$_severe);
477 0586
478 0587 dump$gl_flags[dump$v_tpa_start] = false; ! Clear flags for next call
479 0588 dump$gl_flags[dump$v_tpa_end] = false;
480 0589 dump$gl_flags[dump$v_tpa_count] = false;
481 0590 dump$gl_flags[dump$v_tpa_number] = false;
482 0591
483 0592
484 0593
485 0594 RETURN true ! Return success to TPARSE
486 0595 END;
```

```

                                000C 0000 DUMP$STORE_NUM:
                                .WORD Save R2,R3
                                MOVAB TPA_BLOCK+28, R3
                                MOVAB DUMP$GL_FLAGS, R2
                                BBC #4, DUMP$GL_FLAGS+3, 1$
                                MCVL TPA_BLOCK+28, DUMP$GL_START_QUAL
                                BISB2 #128, DUMP$GL_FLAGS+1
                                BRB 5$
                                BBC #5, DUMP$GL_FLAGS+3, 2$
                                MCVL TPA_BLOCK+28, DUMP$GL_END_QUAL
                                BISB2 #1, DUMP$GL_FLAGS+2
                                BRB 5$
                                0B 03 A2 53 00000000 EF 9E 00002
                                04 A2 52 000C0000 EF 9E 00009
                                01 A2 80 63 D0 00010
                                0A 03 A2 8F 88 00019
                                08 A2 36 11 0001E 1$:
                                02 A2 05 E1 00020
                                01 88 00025
                                27 11 00029
                                01 88 00029
                                27 11 0002D
                                0556
                                0563
                                0566
                                0567
                                0563
                                0569
                                0572
                                0573
                                0569
```


DUMPSMAIN
V04-000

K 11
16-Sep-1984 01:26:41
14-Sep-1984 12:21:35

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[DUMP.SRC]DUMP.B32;1

Page 22
(7)

0A	03	A2	06	E1	0002F	2\$:	BBC	#6, DUMP\$GL_FLAGS+3, 3\$..	0575
	0C	A2	63	D0	00034		MOVL	TPA_BLOCK+28, DUMP\$GL_COUNT_QUAL	..	0578
	02	A2	02	88	00038		BISB2	#2, DUMP\$GL_FLAGS+2	..	0579
			18	11	0003C		BRB	5\$..	0575
			A2	95	0003E	3\$:	TSTB	DUMP\$GL_FLAGS+3	..	0581
			06	18	00041		BGEQ	4\$..	
	10	A2	63	D0	00043		MOVL	TPA_BLOCK+28, DUMP\$GL_NUMBER_QUAL	..	0583
			0D	11	00047		BRB	5\$..	
			8F	DD	00049	4\$:	PUSHL	#<<<DUMP\$ FACILITY@16>+4384>+4>	..	0585
00000000G	00		01	FB	0004F		CALLS	#1, LIB\$STOP	..	
	03		8F	8A	00056	5\$:	BICB2	#240, DUMP\$GL_FLAGS+3	..	0591
			01	D0	0005B		MOVL	#1, R0	..	0594
			04	0005E			RET		..	0595

; Routine Size: 95 bytes, Routine Base: \$CODE\$ + 0366

```
488 0596 1 ROUTINE dump$open_input(fab, filedesc)=
489 0597 BEGIN
490 0598
491 0599 This routine opens the input file
492 0600
493 0601 Inputs:
494 0602     fab      pointer to an already initialized fab complete with NAM block
495 0603     filedesc pointer to string descriptor of resultant string from $parse
496 0604
497 0605
498 0606 Outputs:
499 0607     File is opened
500 0608
501 0609 Routine value:
502 0610
503 0611     true      successful
504 0612     false     error, signal already done
505 0613
506 0614
507 0615 MAP
508 0616     fab : REF BBLOCK,
509 0617     filedesc : REF BBLOCK;
510 0618 LOCAL
511 0619     ixab : $XABFHC DECL,
512 0620     dib : BBLOCK[dTb$c (length),
513 0621     dibdesc : BBLOCK[dSc$c_s_b(n),
514 0622     status;
515 0623
516 0624
517 0625 dump$gl_ifab = .fab; ! Set pointer to FAB
518 0626 dump$gl_inam = .fab[fab$l_nam]; ! and NAM block
519 0627
520 0628 fab[fab$b_shr]=fab$m_get OR fab$m_put OR fab$m_upi; ! Open file shared.
521 0629
522 0630 IF .BBLOCK[fab[fab$l_dev], dev$v_net] ! If network device
523 0631 THEN
524 0632 BEGIN
525 0633     IF .dump$gl_flags[dump$v_allocated] ! Ensure no conflicting
526 0634     OR .dump$gl_flags[dump$v_blocks] ! qualifiers
527 0635     OR .dump$gl_flags[dump$v_header]
528 0636 THEN
529 0637     SIGNAL_STOP(dump$_devquals);
530 0638
531 0639
532 0640 dump$gl_flags[dump$v_records] = true; ! Force record mode
533 0641 END;
534 0642
535 0643
536 0644 IF .BBLOCK[fab[fab$l_dev], dev$v_for] ! If foreign device
537 0645 OR (NOT .BBLOCK[fab[fab$l_dev], dev$v_fod] ! or not disk, tape, or network
538 0646 AND NOT .BBLOCK[fab[fab$l_dev], dev$v_net])
539 0647 THEN
540 0648 BEGIN
541 0649     IF .dump$gl_flags[dump$v_allocated] ! Ensure no file-oriented
542 0650     OR .dump$gl_flags[dump$v_records] ! qualifiers
543 0651     OR .dump$gl_flags[dump$v_header]
544 0652 THEN
```

```
545 0653 SIGNAL_STOP(dump$devquals);
546 0654
547 0655
548 0656 IF (.dump$gl_inam[nam$l_fnb] AND ! Ensure nothing except device
549 0657 (nam$m_exp_dir OR
550 0658 nam$m_exp_name OR
551 0659 nam$m_exp_type OR
552 0660 nam$m_exp_ver OR
553 0661 nam$m_wildcard)) NEQ 0
554 0662 THEN
555 0663 SIGNAL_STOP(dump$devspec);
556 0664 END
557 0665 ELSE
558 0666 BEGIN
559 0667 IF NOT .BBLOCK[fab[fab$l_dev], dev$v_rnd] ! Ensure no disk-oriented
560 0668 AND .dump$gl_flags[dump$v_allocated] ! qualifiers on tape
561 0669 THEN
562 0670 SIGNAL_STOP(dump$devquals);
563 0671
564 0672
565 P 0673 $XABFHC_INIT(xab=ixab, ! Initialize XAB
566 0674 nxt=0);
567 0675 fab[fab$l_xab] = ixab; ! Set pointer to XAB
568 0676 END;
569 0677
570 0678
571 0679 IF NOT .dump$gl_flags[dump$v_records]
572 0680 THEN
573 0681 fab[fab$v_ufo] = true ! Open file only
574 0682 ELSE
575 0683 fab[fab$v_get] = fab[fab$v_sqo] = true; ! Allow GETs, sequential op
576 0684
577 0685
578 0686 IF NOT .BBLOCK[fab[fab$l_dev], dev$v_for] ! Do OPEN if not foreign
579 0687 THEN
580 0688 BEGIN
581 0689 IF NOT $OPEN(fab=.fab) ! Open the input file
582 0690 THEN
583 0691 BEGIN
584 0692 dump$file_error(
585 0693 dump$facility*16 + shr$openin + sts$sk_error,
586 0694 .fab,
587 0695 .fab[fab$l_sts], .fab[fab$l_stv]);
588 0696 fab[fab$l_xab] = 0;
589 0697 RETURN false;
590 0698 END;
591 0699 END;
592 0700
593 0701 fab[fab$l_xab] = 0;
594 0702
595 0703 IF .BBLOCK[fab[fab$l_dev], dev$v_for] ! If foreign device
596 0704 OR (NOT .BBLOCK[fab[fab$l_dev], dev$v_fod] ! or not disk, tape, or network
597 0705 AND NOT .BBLOCK[fab[fab$l_dev], dev$v_net])
598 0706 THEN
599 0707 BEGIN
600 0708 dump$gl_idesc[dsc$v_length] = .dump$gl_inam[nam$b_dev]; ! Prune to
601 0709 dump$gl_idesc[dsc$a_pointer] = .dump$gl_inam[nam$l_dev]; ! device only
```



```
602 P 0710      status = $ASSIGN(DEVNAM = dump$gl_idesc,      ! Do ASSIGN if foreign
603      0711      CHAN = fab[fab$l_stv]);
604      0712      IF NOT .status THEN SIGNAL_STOP(.status);
605      0713      END
606      0714      ELSE
607      0715      BEGIN
608      0716      dump$gl_idesc[dsc$w_length] = .dump$gl_inam[nam$b_rsl];
609      0717      dump$gl_idesc[dsc$a_pointer] = .dump$gl_inam[nam$_rsa];
610      0718      END;
611      0719
612      0720      IF NOT .dump$gl_flags[dump$v_records]
613      0721      THEN
614      0722      dump$gl_channel = .fab[fab$l_stv]      ! Save the channel
615      0723      ELSE
616      0724      BEGIN
617      0725      $RAB_INIT(rab=dump$gl_irab,      ! Initialize input RAB
618      0726      fab=.fab);
619      0727      IF NOT $CONNECT(rab=dump$gl_irab)      ! Connect RAB
620      0728      THEN
621      0729      BEGIN
622      0730      dump$file_error(
623      0731      dump$_facility*16 + shr$_openin + sts$_error,
624      0732      .fab,
625      0733      .dump$gl_irab[rab$l_sts], .dump$gl_irab[rab$l_stv]);
626      0734      RETURN false;
627      0735      END;
628      0736      END;
629      0737
630      0738      dump$gl_cur_block = 1;
631      0739      dump$gl_max_block = -1;
632      0740
633      0741      IF .BBLOCK[fab[fab$l_dev], dev$v_rnd]      ! Disk device
634      0742      THEN
635      0743      IF .BBLOCK[fab[fab$l_dev], dev$v_for]      ! If foreign disk
636      0744      THEN
637      0745      BEGIN
638      0746      dibdesc[dsc$w_length] = dib$_length;      ! Set up to get device
639      0747      dibdesc[dsc$a_pointer] = dib;      ! characteristics
640      0748      status = $GETCHN(CHAN=.dump$gl_channel, PRIBUF=dibdesc);
641      0749      IF NOT .status
642      0750      THEN
643      0751      SIGNAL_STOP(dump$_getchn, 0, .status);
644      0752      dump$gl_cur_block = 0;
645      0753      dump$gl_max_block = .dib[dib$_maxblock] - 1;
646      0754      END
647      0755      ELSE
648      0756      BEGIN      ! Files-11 disk
649      0757      ! Save FHC information for page heading.
650      0758      dump$gl_file_efblk = .ixab[xab$l_ebk];
651      0759      IF .dump$gl_file_efblk NEQ 0 AND .ixab[xab$w_ffb] EQL 0
652      0760      THEN
653      0761      dump$gl_file_efblk = .dump$gl_file_efblk - 1;
654      0762      dump$gl_file_hiblk = .fab[fab$l_atq];
655      0763
656      0764
657      0765
658      0766
```

```
659 0767 3      IF NOT .dump$gl_flags[dump$sv_records]      ! Not record mode
660 0768 3      THEN
661 0769 3          BEGIN
662 0770 3              dump$gl_max_block = .dump$gl_file_efblk;
663 0771 3              IF .dump$gl_flags[dump$sv_allocated]
664 0772 3                  THEN dump$gl_max_block = .dump$gl_file_hiblk;
665 0773 3              END;
666 0774 3          END;
667 0775 3
668 0776 3      IF .dump$gl_flags[dump$sv_start]
669 0777 3      THEN
670 0778 3          dump$gl_cur_block = MAXU(.dump$gl_cur_block, .dump$gl_start_qual);
671 0779 3
672 0780 3      IF .BBLOCK[fab[fab$l_dev], dev$sv_for]
673 0781 3      AND .dump$gl_cur_block GTRU .dump$gl_max_block
674 0782 3      THEN SIGNAL_STOP(dump$_badstart, 1, .dump$gl_max_block);
675 0783 3
676 0784 3      IF .dump$gl_flags[dump$sv_end]
677 0785 3      THEN
678 0786 3          dump$gl_max_block = MINU(.dump$gl_max_block, .dump$gl_end_qual);
679 0787 3
680 0788 3      IF .dump$gl_flags[dump$sv_count]
681 0789 3      THEN
682 0790 3          IF .dump$gl_flags[dump$sv_start]
683 0791 3          THEN
684 0792 3              dump$gl_max_block = MINU(.dump$gl_max_block,
685 0793 3                  .dump$gl_start_qual + .dump$gl_count_qual - 1)
686 0794 3          ELSE
687 0795 3              dump$gl_max_block = MINU(.dump$gl_max_block,
688 0796 3                  .dump$gl_cur_block + .dump$gl_count_qual - 1);
689 0797 3
690 0798 3      dump$gl_record = 0;
691 0799 3      IF NOT .dump$gl_flags[dump$sv_records]
692 0800 3      AND .BBLOCK[fab[fab$l_dev], dev$sv_rnd]
693 0801 3      THEN
694 0802 3          dump$gl_record = .dump$gl_cur_block - 1;
695 0803 3
696 0804 3
697 0805 3      ! Allocate input buffer.
698 0806 3
699 0807 3
700 0808 3      IF .dump$gl_flags[dump$sv_records]      ! If record dump
701 0809 3      THEN
702 0810 3          dump$gl_buffer[dsc$w_length] = dump$c_rmsbufsz      ! Largest RMS record
703 0811 3      ELSE
704 0812 3          IF .BBLOCK[fab[fab$l_dev], dev$sv_sqd]
705 0813 3          THEN
706 0814 3              dump$gl_buffer[dsc$w_length] = dump$c_tapbufsz      ! Largest tape QIO
707 0815 3          ELSE
708 0816 3              dump$gl_buffer[dsc$w_length] = dump$c_qiobufsz;      ! Largest non-tape QIO
709 0817 3
710 0818 3
711 0819 3      status = lib$get_vm(      ! Get memory
712 0820 3          dump$gl_buffer[dsc$w_length],
713 0821 3          dump$gl_buffer[dsc$a_pointer]);
714 0822 3      IF NOT .status THEN SIGNAL_STOP(dump$_novirmem, 0, .status);
715 0823 3
```

```
SRMS_PTR=          DUMPSGL IRAB
                  .EXTRN SYSS$OPEN, SYSS$ASSIGN
                  .EXTRN SYSS$CONNECT, SYSS$GETCHN
```

WORD Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11

PC	Op	OpC	OpD	OpI	OpR	OpS	OpT	OpV	OpW	OpX	OpY	OpZ	OpAA	OpAB	OpAC	OpAD	OpAE	OpAF	OpAG	OpAH	OpAI	OpAJ	OpAK	OpAL	OpAM	OpAN	OpAO	OpAP	OpAQ	OpAR	OpAS	OpAT	OpAU	OpAV	OpAW	OpAX	OpAY	OpAZ	OpBA	OpBB	OpBC	OpBD	OpBE	OpBF	OpBG	OpBH	OpBI	OpBJ	OpBK	OpBL	OpBM	OpBN	OpBO	OpBP	OpBQ	OpBR	OpBS	OpBT	OpBU	OpBV	OpBW	OpBX	OpBY	OpBZ	OpCA	OpCB	OpCC	OpCD	OpCE	OpCF	OpCG	OpCH	OpCI	OpCJ	OpCK	OpCL	OpCM	OpCN	OpCO	OpCP	OpCQ	OpCR	OpCS	OpCT	OpCU	OpCV	OpCW	OpCX	OpCY	OpCZ	OpDA	OpDB	OpDC	OpDD	OpDE	OpDF	OpDG	OpDH	OpDI	OpDJ	OpDK	OpDL	OpDM	OpDN	OpDO	OpDP	OpDQ	OpDR	OpDS	OpDT	OpDU	OpDV	OpDW	OpDX	OpDY	OpDZ	OpEA	OpEB	OpEC	OpED	OpEE	OpEF	OpEG	OpEH	OpEI	OpEJ	OpEK	OpEL	OpEM	OpEN	OpEO	OpEP	OpEQ	OpER	OpES	OpET	OpEU	OpEV	OpEW	OpEX	OpEY	OpEZ	OpFA	OpFB	OpFC	OpFD	OpFE	OpFF	OpFG	OpFH	OpFI	OpFJ	OpFK	OpFL	OpFM	OpFN	OpFO	OpFP	OpFQ	OpFR	OpFS	OpFT	OpFU	OpFV	OpFW	OpFX	OpFY	OpFZ	OpGA	OpGB	OpGC	OpGD	OpGE	OpGF	OpGG	OpGH	OpGI	OpGJ	OpGK	OpGL	OpGM	OpGN	OpGO	OpGP	OpGQ	OpGR	OpGS	OpGT	OpGU	OpGV	OpGW	OpGX	OpGY	OpGZ	OpHA	OpHB	OpHC	OpHD	OpHE	OpHF	OpHG	OpHH	OpHI	OpHJ	OpHK	OpHL	OpHM	OpHN	OpHO	OpHP	OpHQ	OpHR	OpHS	OpHT	OpHU	OpHV	OpHW	OpHX	OpHY	OpHZ	OpIA	OpIB	OpIC	OpID	OpIE	OpIF	OpIG	OpIH	OpII	OpIJ	OpIK	OpIL	OpIM	OpIN	OpIO	OpIP	OpIQ	OpIR	OpIS	OpIT	OpIU	OpIV	OpIW	OpIX	OpIY	OpIZ	OpJA	OpJB	OpJC	OpJD	OpJE	OpJF	OpJG	OpJH	OpJI	OpJJ	OpJK	OpJL	OpJM	OpJN	OpJO	OpJP	OpJQ	OpJR	OpJS	OpJT	OpJU	OpJV	OpJW	OpJX	OpJY	OpJZ	OpKA	OpKB	OpKC	OpKD	OpKE	OpKF	OpKG	OpKH	OpKI	OpKJ	OpKK	OpKL	OpKM	OpKN	OpKO	OpKP	OpKQ	OpKR	OpKS	OpKT	OpKU	OpKV	OpKW	OpKX	OpKY	OpKZ	OpLA	OpLB	OpLC	OpLD	OpLE	OpLF	OpLG	OpLH	OpLI	OpLJ	OpLK	OpLL	OpLM	OpLN	OpLO	OpLP	OpLQ	OpLR	OpLS	OpLT	OpLU	OpLV	OpLW	OpLX	OpLY	OpLZ	OpMA	OpMB	OpMC	OpMD	OpME	OpMF	OpMG	OpMH	OpMI	OpMJ	OpMK	OpML	OpMM	OpMN	OpMO	OpMP	OpMQ	OpMR	OpMS	OpMT	OpMU	OpMV	OpMW	OpMX	OpMY	OpMZ	OpNA	OpNB	OpNC	OpND	OpNE	OpNF	OpNG	OpNH	OpNI	OpNJ	OpNK	OpNL	OpNM	OpNN	OpNO	OpNP	OpNQ	OpNR	OpNS	OpNT	OpNU	OpNV	OpNW	OpNX	OpNY	OpNZ	OpOA	OpOB	OpOC	OpOD	OpOE	OpOF	OpOG	OpOH	OpOI	OpOJ	OpOK	OpOL	OpOM	OpON	OpOO	OpOP	OpOQ	OpOR	OpOS	OpOT	OpOU	OpOV	OpOW	OpOX	OpOY	OpOZ	OpPA	OpPB	OpPC	OpPD	OpPE	OpPF	OpPG	OpPH	OpPI	OpPJ	OpPK	OpPL	OpPM	OpPN	OpPO	OpPP	OpPQ	OpPR	OpPS	OpPT	OpPU	OpPV	OpPW	OpPX	OpPY	OpPZ	OpQA	OpQB	OpQC	OpQD	OpQE	OpQF	OpQG	OpQH	OpQI	OpQJ	OpQK	OpQL	OpQM	OpQN	OpQO	OpQP	OpQQ	OpQR	OpQS	OpQT	OpQU	OpQV	OpQW	OpQX	OpQY	OpQZ	OpRA	OpRB	OpRC	OpRD	OpRE	OpRF	OpRG	OpRH	OpRI	OpRJ	OpRK	OpRL	OpRM	OpRN	OpRO	OpRP	OpRQ	OpRR	OpRS	OpRT	OpRU	OpRV	OpRW	OpRX	OpRY	OpRZ	OpSA	OpSB	OpSC	OpSD	OpSE	OpSF	OpSG	OpSH	OpSI	OpSJ	OpSK	OpSL	OpSM	OpSN	OpSO	OpSP	OpSQ	OpSR	OpSS	OpST	OpSU	OpSV	OpSW
----	----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------

	25	03	A7	E8	000B3	11\$:	BLBS	3(R7), 12\$	0686
			56	DD	000B7		PUSHL	R6	0689
00000000G	00		01	FB	000B9		CALLS	#1, SYSSOPEN	
	19		50	E8	000C0		BLBS	R0, 12\$	
	7E	08	A6	7D	000C3		MOVQ	8(R6), -(SP)	0695
			56	DD	000C7		PUSHL	R6	0694
00000000V	EF	00000000*	8F	DD	000C9		PUSHL	#<<<DUMPS FACILITY@16>+4248>+2>	0693
			04	FB	000CF		CALLS	#4, DUMPSFILE_ERROR	
		24	A6	D4	000D6		CLRL	36(R6)	0696
			01D8	31	000D9		BRW	37\$	0697
		24	A6	D4	000DC	12\$:	CLRL	36(R6)	0701
	50	FD14	C9	D0	000DF		MOVL	DUMPSGL_INAM, R0	0708
2D	08	03	A7	E8	000E4		BLBS	3(R7), 13\$	0703
29	67		0E	E0	000E8		BBS	#14, (R7), 14\$	0704
	67		0D	E0	000EC		BBS	#13, (R7), 14\$	0705
FF50	C9	39	A0	9B	000F0	13\$:	MOVZBW	57(R0), DUMPSGL_IDESC	0708
FF54	C9	44	A0	D0	000F6		MOVL	68(R0), DUMPSGL_IDESC+4	0709
			7E	7C	000FC		CLRQ	-(SP)	0711
		0C	A6	9F	000FE		PUSHAB	12(R6)	
		FF50	C9	9F	00101		PUSHAB	DUMPSGL_IDESC	
00000000G	00		04	FB	00105		CALLS	#4, SYSSASSIGN	
	58		50	D0	0010C		MOVL	R0, STATUS	
	13		58	E8	0010F		BLBS	STATUS, 15\$	0712
			58	DD	00112		PUSHL	STATUS	
	6A		01	FB	00114		CALLS	#1, LIB\$STOP	
			0C	11	00117		BRB	15\$	0703
FF50	C9	03	A0	9B	00119	14\$:	MOVZBW	3(R0), DUMPSGL_IDESC	0716
FF54	C9	04	A0	D0	0011F		MOVL	4(R0), DUMPSGL_IDESC+4	0717
07	01	A9	05	E0	00125	15\$:	BBS	#5, DUMPSGL_FLAGS+1, 16\$	0721
EC	A9	0C	A6	D0	0012A		MOVL	12(R6), DUMPSGL_CHANNEL	0723
			3B	11	0012F		BRB	17\$	
0044	8F	00	00	2C	00131	16\$:	MOVCS	#0, (SP), #0, #68, \$RMS_PTR	0727
			00		00138				
FD18	C9	FD18	C9	8F	B0	0013B	MOVW	#17409, \$RMS_PTR	
FD54	C9	4401	56	D0	00142		MOVL	R6, \$RMS_PTR+60	
			C9	9F	00147		PUSHAB	DUMPSGL_IRAB	0728
00000000G	00	FD18	01	FB	0014B		CALLS	#1, SYSSCONNECT	
	17		50	E8	00152		BLBS	R0, 17\$	
	7E	FD20	C9	7D	00155		MOVQ	DUMPSGL_IRAB+8, -(SP)	0734
			56	DD	0015A		PUSHL	R6	0733
		00000000*	8F	DD	0015C		PUSHL	#<<<DUMPS FACILITY@16>+4248>+2>	0732
00000000V	EF		04	FB	00162		CALLS	#4, DUMPSFILE_ERROR	
			0148	31	00169		BRW	37\$	0735
	18	A9	01	D0	0016C	17\$:	MOVL	#1, DUMPSGL_CUR_BLOCK	0740
	1C	A9	01	CE	00170		MNEGL	#1, DUMPSGL_MAX_BLOCK	0741
62	67		1C	E1	00174		BBC	#28, (R7), 21\$	0743
	38	03	A7	E9	00178		BLBC	3(R7), 19\$	0745
	6E	74	8F	9B	0017C		MOVZBW	#116, DIBDESC	0748
	04	08	AE	9E	00180		MOVAB	DIB, DIBDESC+4	0749
			7E	7C	00185		CLRQ	-(SP)	0750
		08	AE	9F	00187		PUSHAB	DIBDESC	
			7E	D4	0018A		CLRL	-(SP)	
		EC	A9	DD	0018C		PUSHL	DUMPSGL_CHANNEL	
00000000G	00		05	FB	0018F		CALLS	#5, SYSSGETCHN	
	58		50	D0	00196		MOVL	R0, STATUS	
	0D		58	E8	00199		BLBS	STATUS, 18\$	0751
			58	DD	0019C		PUSHL	STATUS	0753

			6A	00000000G	7E	D4	0019E	CLRL	-(SP)			
					8F	DD	001A0	PUSHL	#DUMPS GETCHN			
					03	FB	001A6	CALLS	#3, LIB\$STOP			
1C	A9	78	AE	18	A9	D4	001A9	CLRL	DUMPSGL_CUR_BLOCK	0754		
					01	C3	001AC	SUBL3	#1, DIB7112, DUMPSGL_MAX_BLOCK	0755		
		20	A9	E4	26	11	001B2	BRB	21\$	0745		
					AD	D0	001B4	MOVL	IXAB+16, DUMPSGL_FILE_EFBLK	0762		
					08	13	001B9	BEQL	20\$	0763		
					AD	B5	001BB	TSTW	IXAB+20			
					03	12	001BE	BNEQ	20\$			
					20	A9	D7	001C0	DECL	DUMPSGL_FILE_EFBLK	0765	
		24	A9	10	A6	D0	001C3	MOVL	16(R6), DUMPSGL_FILE_HIBLK	0766		
0D		01	A9		05	E0	001C8	BBS	#5, DUMPSGL_FLAGS+1, 21\$	0767		
		1C	A9	20	A9	D0	001CD	MOVL	DUMPSGL_FILE_EFBLK, DUMPSGL_MAX_BLOCK	0770		
					69	E9	001D2	BLBC	DUMPSGL_FLAGS, 21\$	0771		
		1C	A9	24	A9	D0	001D5	MOVL	DUMPSGL_FILE_HIBLK, DUMPSGL_MAX_BLOCK	0772		
					01	A9	95	001DA	TSTB	DUMPSGL_FLAGS+1	0776	
					12	18	001DD	BGEQ	23\$			
			50	18	A9	D0	001DF	MOVL	DUMPSGL_CUR_BLOCK, R0	0778		
		04	A9		50	D1	001E3	CMPL	R0, DUMPSGL_START_QUAL			
					04	1E	001E7	BGEQU	22\$			
			50	04	A9	D0	001E9	MOVL	DUMPSGL_START_QUAL, R0			
		18	A9		50	D0	001ED	MOVL	R0, DUMPSGL_CUR_BLOCK			
			15	03	A7	E9	001F1	BLBC	3(R7), 24\$	0780		
		1C	A9	18	A9	D1	001F5	CMPL	DUMPSGL_CUR_BLOCK, DUMPSGL_MAX_BLOCK	0781		
					0E	1B	001FA	BLEQU	24\$			
					1C	A9	DD	001FC	PUSHL	DUMPSGL_MAX_BLOCK	0782	
					01	DD	001FF	PUSHL	#1			
			6A	00000000G	8F	DD	00201	PUSHL	#DUMPS BADSTART			
			12	02	03	FB	00207	CALLS	#3, LIB\$STOP			
			50	1C	A9	E9	0020A	BLBC	DUMPSGL_FLAGS+2, 26\$	0784		
		08	A9		A9	D0	0020E	MOVL	DUMPSGL_MAX_BLOCK, R0	0786		
					50	D1	00212	CMPL	R0, DUMPSGL_END_QUAL			
			50	08	04	1B	00216	BLEQU	25\$			
		1C	A9		A9	D0	00218	MOVL	DUMPSGL_END_QUAL, R0			
30		02	A9		50	D0	0021C	MOVL	R0, DUMPSGL_MAX_BLOCK			
					01	E1	00220	BBC	#1, DUMPSGL_FLAGS+2, 30\$	0788		
					A9	95	00225	TSTB	DUMPSGL_FLAGS+1	0790		
					13	18	00228	BGEQ	27\$			
51		04	A9	0C	A9	C1	0022A	ADDL3	DUMPSGL_COUNT_QUAL, DUMPSGL_START_QUAL, R1	0793		
					51	D7	00230	DECL	R1			
			50	1C	A9	D0	00232	MOVL	DUMPSGL_MAX_BLOCK, R0			
			51		50	D1	00236	CMPL	R0, R1			
					13	1A	00239	BGTRU	28\$			
					14	11	0023B	BRB	29\$	0792		
51		18	A9	0C	A9	C1	0023D	ADDL3	DUMPSGL_COUNT_QUAL, DUMPSGL_CUR_BLOCK, R1	0796		
					51	D7	00243	DECL	R1			
			50	1C	A9	D0	00245	MOVL	DUMPSGL_MAX_BLOCK, R0			
			51		50	D1	00249	CMPL	R0, R1			
					03	1B	0024C	BLEQU	29\$			
			50		51	D0	0024E	MOVL	R1, R0			
		1C	A9		50	D0	00251	MOVL	R0, DUMPSGL_MAX_BLOCK	0795		
					A9	D4	00255	CLRL	DUMPSGL_RECORD	0799		
					05	E0	00258	BBS	#5, DUMPSGL_FLAGS+1, 32\$	0800		
28	0F	01	A9		1C	E1	0025D	BBC	#28, (R7), 31\$	0801		
	06	18	A9		01	C3	00261	SUBL3	#1, DUMPSGL_CUR_BLOCK, DUMPSGL_RECORD	0803		
	08	01	A9		05	E1	00267	BBC	#5, DUMPSGL_FLAGS+1, 33\$	0808		

DUMPSMAIN
V04-000

F 12
16-Sep-1984 01:26:41
14-Sep-1984 12:21:35

VAX-11 BLISS-32 V4.0-742
DISK\$VMSMASTER:[DUMP.SRC]DUMP.B32;1

Page 30
(8)

06	F8	A9	7FFF	8F	B0	0026C	32\$:	MOVW	#32767, DUMPSGL_BUFFER	0810
				10	11	00272		BRB	35\$	
		67		05	E1	00274	33\$:	BBC	#5, (R7), 34\$	0812
	F8	A9		01	AE	00278		MNEGW	#1, DUMPSGL_BUFFER	0814
				06	11	0027C		BRB	35\$	
	F8	A9	0200	8F	B0	0027E	34\$:	MOVW	#512, DUMPSGL_BUFFER	0816
			FC	A9	9F	00284	35\$:	PUSHAB	DUMPSGL_BUFFER+4	0821
			F8	A9	9F	00287		PUSHAB	DUMPSGL_BUFFER	0820
00000000G		00		02	FB	0028A		CALLS	#2, LIB\$GET_VM	
		58		50	D0	00291		MOVL	R0, STATUS	
		0D		58	E8	00294		BLBS	STATUS, 36\$	0822
				58	DD	00297		PUSHL	STATUS	
				7E	D4	00299		CLRL	-(SP)	
			00000000G	8F	DD	0029B		PUSHL	#DUMPS NOVIRMEM	
		6A		03	FB	002A1		CALLS	#3, LIB\$STOP	
FD38	C9		F8	A9	B0	002A4	36\$:	MOVW	DUMPSGL_BUFFER, DUMPSGL_IRAB+32	0825
FD3C	C9		FC	A9	D0	002AA		MOVL	DUMPSGL_BUFFER+4, DUMPSGL_IRAB+36	0826
		50		01	D0	002B0		MOVL	#1, R0	0827
					04	002B3		RET		
				50	D4	002B4	37\$:	CLRL	R0	0828
				04	002B6			RET		

; Routine Size: 695 bytes. Routine Base: \$CODE\$ + 03C5


```
0829 1 ROUTINE dump$open_output(output_desc, ifab)=
0830 BEGIN
0831
0832 Open output file
0833
0834 Inputs:
0835
0836 output_desc pointer to string descriptor for output file
0837 ifab pointer to input fab
0838
0839 MAP
0840 output_desc : REF BBLOCK,
0841 ifab : REF BBLOCK;
0842
0843
0844 $FAB_INIT(fab=dump$gl_ofab, ! Initialize output FAB
0845 dna=UPLIT BYTE('DMP'), ! Default /OUTPUT type
0846 dns=CHARCOUNT('DMP'),
0847 nam=dump$gl_onam,
0848 fop=<ofp,sqo>,
0849 rat=cr,
0850 fac=put);
0851
0852
0853 $NAM_INIT(nam=dump$gl_onam, ! Initialize output NAM block
0854 flf=.ifab[fab$l_nam],
0855 rss=nam$c_maxrss,
0856 rsa=dump$gl_orss,
0857 ess=nam$c_maxrss,
0858 esa=dump$gl_orss);
0859
0860
0861 $RAB_INIT(rab=dump$gl_orab, ! Initialize output RAB
0862 fab=dump$gl_ofab);
0863
0864
0865 ! Create the output file and connect record stream.
0866
0867 IF .dump$gl_flags[dump$v_printer] ! If /PRINTER requested,
0868 THEN
0869 BEGIN
0870 dump$gl_ofab[fab$v_spl] = true; ! Spool listing
0871 dump$gl_ofab[fab$v_dlt] = true; ! Delete after printing
0872 END
0873 ELSE
0874 IF .dump$gl_flags[dump$v_output] ! If /OUTPUT requested
0875 THEN
0876 BEGIN
0877 IF .output_desc[dsc$w_length] NEQ 0 ! If /OUTPUT has a value
0878 THEN
0879 BEGIN
0880 dump$gl_ofab[fab$l_fna] = .output_desc[dsc$a_pointer];
0881 dump$gl_ofab[fab$b_fns] = .output_desc[dsc$w_length];
0882 END
0883 END
0884 ELSE
0885 BEGIN ! Else, default to SYS$OUTPUT
```

```

779      0886      dump$gl_ofab[fab$l_fna] = UPLIT BYTE('SYSS$OUTPUT');
780      0887      dump$gl_ofab[fab$b_fns] = %CHARCOUNT('SYSS$OUTPUT');
781      0888      END;
782      0889
783      0890
784      0891      IF NOT $CREATE(fab=dump$gl_ofab)
785      0892      THEN
786      0893      BEGIN
787      0894      dump$file_error(
788      0895      dump$facility*16 + shr$_openout + sts$_error,
789      0896      dump$gl_ofab,
790      0897      .dump$gl_ofab[fab$l_sts], .dump$gl_ofab[fab$l_stv]);
791      0898      RETURN false;
792      0899      END;
793      0900
794      0901      IF NOT $CONNECT(rab=dump$gl_orab)
795      0902      THEN
796      0903      BEGIN
797      0904      dump$file_error(
798      0905      dump$facility*16 + shr$_openout + sts$_error,
799      0906      dump$gl_ofab,
800      0907      .dump$gl_orab[rab$l_sts], .dump$gl_orab[rab$l_stv]);
801      0908      RETURN false;
802      0909      END;
803      0910
804      0911
805      0912      dump$gl_odesc[dsc$_length] = .dump$gl_onam[nam$b_rsl];
806      0913      dump$gl_odesc[dsc$_pointer] = .dump$gl_onam[nam$_rsa];
807      0914      RETURN true;
808      0915      END;

```

.PSECT \$SPLITS,NOWRT,NOEXE,2

54 55 50 54 55 4F 50 4D 44 2E 00158 P.ABQ: .ASCII \.DMP\
24 53 59 53 0015C P.ABR: .ASCII \SYSS\$OUTPUT\

\$RMS_PTR= DUMP\$GL_OFAB
\$RMS_PTR= DUMP\$GL_ONAM
\$RMS_PTR= DUMP\$GL_ORAB
.EXTRN SYSS\$CREATE

.PSECT \$CODE\$,NOWRT,2

007C 00000 DUMPSOPEN OUTPUT:

0050	8F	00	56	00000000'	EF	9E	00002	WORD	Save R2,R3,R4,R5,R6	0829
			6E		00	2C	00009	MOVAB	\$RMS_PTR, R6	0850
					66		00010	MOVCS	#0, TSP), #0, #80, \$RMS_PTR	
			66	5003	8F	80	00011	MOVW	#20483, \$RMS_PTR	
		04	A6	20000040	8F	80	00016	MOVL	#536870976, \$RMS_PTR+4	
		16	A6		01	90	0001E	MOVB	#1, \$RMS_PTR+22	
		1E	A6	0202	8F	80	00022	MOVW	#514, \$RMS_PTR+30	
		28	A6	50	A6	9E	00028	MOVAB	DUMP\$GL_ONAM, \$RMS_PTR+40	
		30	A6	00000000'	EF	9E	0002D	MOVAB	P.ABQ, \$RMS_PTR+48	
		35	A6		04	90	00035	MOVB	#4, \$RMS_PTR+53	

; Routine Size: 241 bytes, Routine Base: \$CODES + 067C

```
0010 810 0916 1 GLOBAL ROUTINE dump$read(bufdesc)=
0011 811 0917 BEGIN
0012 812 0918
0013 813 0919
0014 814 0920
0015 815 0921
0016 816 0922
0017 817 0923
0018 818 0924
0019 819 0925
0020 820 0926
0021 821 0927
0022 822 0928
0023 823 0929
0024 824 0930
0025 825 0931
0026 826 0932
0027 827 0933
0028 828 0934
0029 829 0935
0030 830 0936
0031 831 0937
0032 832 0938
0033 833 0939
0034 834 0940
0035 835 0941
0036 836 0942
0037 837 0943
0038 838 0944
0039 839 0945
0040 840 0946
0041 841 0947
0042 842 0948
0043 843 0949
0044 844 0950
0045 845 0951
0046 846 0952
0047 847 0953
0048 848 0954
0049 849 0955
0050 850 0956
0051 851 0957
0052 852 0958
0053 853 0959
0054 854 0960
0055 855 0961
0056 856 0962
0057 857 0963
0058 858 0964
0059 859 0965
0060 860 0966
0061 861 0967
0062 862 0968
0063 863 0969
0064 864 0970
0065 865 0971
0066 866 0972

P 0916 1 GLOBAL ROUTINE dump$read(bufdesc)=
P 0917 BEGIN
P 0918
P 0919
P 0920
P 0921
P 0922
P 0923
P 0924
P 0925
P 0926
P 0927
P 0928
P 0929
P 0930
P 0931
P 0932
P 0933
P 0934
P 0935
P 0936
P 0937
P 0938
P 0939
P 0940
P 0941
P 0942
P 0943
P 0944
P 0945
P 0946
P 0947
P 0948
P 0949
P 0950
P 0951
P 0952
P 0953
P 0954
P 0955
P 0956
P 0957
P 0958
P 0959
P 0960
P 0961
P 0962
P 0963
P 0964
P 0965
P 0966
P 0967
P 0968
P 0969
P 0970
P 0971
P 0972

1 GLOBAL ROUTINE dump$read(bufdesc)=
BEGIN
This routine reads from the input file.
MAP
bufdesc : REF BBLOCK;
LOCAL
iosb : VECTOR[4,WORD],
status;

IF NOT .dump$gl_flags[dump$vr_records] ! If reading with QIO
THEN
BEGIN
IF NOT .BBLOCK[dump$gl_ifab[fab$l_dev], dev$vr_rnd]
THEN
BEGIN ! One QIO = one block
DECR i FROM 1 TO 0 DO
BEGIN
status = $QIOW(
CHAN=.dump$gl_channel,
FUNC=(IF .BBLOCK[dump$gl_ifab[fab$l_dev], dev$vr_for]
THEN ios_readblk
ELSE ios_readvblk),
IOSB=iosb,
P1=.dump$gl_buffer[dsc$a_pointer],
P2=.dump$gl_buffer[dsc$w_length]);
bufdesc[dsc$w_length] = .iosb[1]; ! Bytes actually read
bufdesc[dsc$a_pointer] = .dump$gl_buffer[dsc$a_pointer];
IF .status THEN status = .iosb[0];

IF .BBLOCK[dump$gl_ifab[fab$l_dev], dev$vr_trm] ! Handle ^Z
AND .iosb[2] EQL '032' ! from terminal
THEN
status = ss$endoffile;

IF .status EQL ss$endoffile ! Print message if end of file
AND .BBLOCK[dump$gl_ifab[fab$l_dev], dev$vr_sqd]
AND .BBLOCK[dump$gl_ifab[fab$l_dev], dev$vr_for]
THEN
BEGIN
dump$blank_line();
dump$output_getmsg(dump$endoffile, '0001');
IF .i EQL 0 THEN EXITLOOP;
END
ELSE
EXITLOOP;
END;
ELSE
END
ELSE
BEGIN
IF .dump$gl_cur_block GTRU .dump$gl_max_block
THEN
RETURN ss$endoffile; ! Return EOF status
status = $QIOW
CHAN=.dump$gl_channel,
```



```
867 P 0973 4 FUNC=(IF ,BBLOCK[dump$gl_ifab[fab$l_dev], dev$v_for]
868 P 0974 4 THEN ios_readblk
869 P 0975 4 ELSE ios_readvblk),
870 P 0976 4 !OSB=iosb,
871 P 0977 4 P1=.dump$gl_buffer[dsc$a_pointer],
872 P 0978 4 P2=512,
873 P 0979 4 P3=.dump$gl_cur_block);
874 P 0980 4 IF .status THEN status = iosb[0];
875 P 0981 4 bufdesc[dsc$w_length] = 512;
876 P 0982 4 bufdesc[dsc$a_pointer] = .dump$gl_buffer[dsc$a_pointer];
877 P 0983 4 dump$gl_cur_block = .dump$gl_cur_block + 1; ! Advance pointer
878 P 0984 4 END;
879 P 0985 4 IF NOT .status
880 P 0986 4 AND .status NEQ ss$_endoffile
881 P 0987 4 AND .status NEQ ss$_parity
882 P 0988 4 AND .status NEQ ss$_datacheck
883 P 0989 4 AND .status NEQ ss$_endoftape
884 P 0990 4 AND .status NEQ ss$_illblknum
885 P 0991 4 THEN
886 P 0992 4 BEGIN
887 P 0993 4 SIGNAL(
888 P 0994 4 dump$_facility*16 + shr$_readerr + sts$_error,
889 P 0995 4 1, dump$gl_idesc,
890 P 0996 4 .status);
891 P 0997 4 status = ss$_endoffile;
892 P 0998 4 END
893 P 0999 4 END
894 P 1000 4 ELSE
895 P 1001 4 BEGIN
896 P 1002 4 status = $GET(rab=dump$gl_irab); ! Get record
897 P 1003 4 bufdesc[dsc$w_length] = .dump$gl_irab[rab$w_rsz];
898 P 1004 4 bufdesc[dsc$a_pointer] = .dump$gl_irab[rab$l_rbf];
899 P 1005 4 IF NOT .status
900 P 1006 4 THEN
901 P 1007 4 BEGIN
902 P 1008 4 IF .status NEQ rms$_eof
903 P 1009 4 THEN
904 P 1010 4 SIGNAL(
905 P 1011 4 dump$_facility*16 + shr$_readerr + sts$_error,
906 P 1012 4 1, dump$gl_idesc,
907 P 1013 4 .dump$gl_irab[rab$l_sts], .dump$gl_irab[rab$l_stv]);
908 P 1014 4 status = ss$_endoffile;
909 P 1015 4 END;
910 P 1016 4 END;
911 P 1017 4
912 P 1018 4
913 P 1019 4 RETURN .status
914 P 1020 4 END;
```

.EXTRN SYSSQIOW, SYSSGET

```
57 00000000G 00 00FC 00000
56 00000000G 00 00 9E 00002
55 00000000G 00 00 9E 00009
55 00000000G EF 00 9E 00010
```

```
.ENTRY DUMPSREAD, Save R2,R3,R4,R5,R6,R7
MOVAB LIB$SIGNAL, R7
MOVAB SYSSQIOW, R6
MOVAB DUMPSGL_BUFFER+4, R5
```

: 0916
:
:
:

		5E		08	C2	00017		SUBL2	#8, SP		
		53		AC	DO	0001A		MOVL	BUFDESC, R3		0945
03	05	A5		05	E1	0001E		BBC	#5, DUMP\$GL_FLAGS+1, 1\$		0928
				0119	31	00023		BRW	14\$		
		50	FD14	C5	DO	00026	1\$:	MOVL	DUMP\$GL_IFAB, R0		0931
03	43	A0		04	E1	00028		BBC	#4, 67(R0), 2\$		
				0081	31	00030		BRW	8\$		
		54		01	DO	00033	2\$:	MOVL	#1, I		0934
				7E	7C	00036	3\$:	CLRQ	-(SP)		0943
				7E	7C	00038		CLRQ	-(SP)		
		7E	FC	A5	3C	0003A		MOVZWL	DUMP\$GL_BUFFER, -(SP)		
				65	DD	0003E		PUSHL	DUMP\$GL_BUFFER+4		
				7E	7C	00040		CLRQ	-(SP)		
				20	AE	9F	00042	PUSHAB	IOSB		
		50	FD14	C5	DO	00045		MOVL	DUMP\$GL_IFAB, R0		
		04	43	A0	E9	0004A		BLBC	67(R0), -4\$		
				21	DD	0004E		PUSHL	#33		
				02	11	00050		BRB	5\$		
				31	DD	00052	4\$:	PUSHL	#49		
			F0	A5	DD	00054	5\$:	PUSHL	DUMP\$GL_CHANNEL		
				7E	D4	00057		CLRL	-(SP)		
		66		0C	FB	00059		CALLS	#12, SYSSQIOW		
		52		50	DO	0005C		MOVL	R0, STATUS		
	04	BC	02	AE	B0	0005F		MOVW	IOSB+2, @BUFDESC		0944
	04	A3		65	DO	00064		MOVL	DUMP\$GL_BUFFER+4, 4(R3)		0945
		03		52	E9	00068		BLBC	STATUS, -6\$		0946
		52		6E	3C	0006B		MOVZWL	IOSB, STATUS		
		50	FD14	C5	DO	0006E	6\$:	MOVL	DUMP\$GL_IFAB, R0		0948
0B	40	A0		02	E1	00073		BBC	#2, 64(R0), 7\$		
		1A		04	AE	B1	00078	CMPW	IOSB+4, #26		0949
				05	12	0007C		BNEQ	7\$		
		52	0870	8F	3C	0007E		MOVZWL	#2160, STATUS		0951
	00000870	8F		52	D1	00083	7\$:	CMPL	STATUS, #2160		0953
				70	12	0008A		BNEQ	13\$		
6B	40	A0		05	E1	0008C		BBC	#5, 64(R0), 13\$		0954
		67	43	A0	E9	00091		BLBC	67(R0), 13\$		0955
	00000000G	00		00	FB	00095		CALLS	#0, DUMP\$BLANK_LINE		0958
				01	DD	0009C		PUSHL	#1		0959
			00000000G	8F	DD	0009E		PUSHL	#DUMP\$ ENDOFFILE		
	00000000G	00		02	FB	000A4		CALLS	#2, DUMP\$OUTPUT_GETMSG		
				54	D5	000AB		TSTL	I		0960
				4D	13	000AD		BEQL	13\$		
		84		54	F4	000AF		SOBGEQ	I, 3\$		0934
				48	11	000B2		BRB	13\$		0931
	20	A5	1C	A5	D1	000B4	8\$:	CMPL	DUMP\$GL_CUR_BLOCK, DUMP\$GL_MAX_BLOCK		0968
				06	1B	000B9		BLEQU	9\$		
		50	0870	8F	3C	000BB		MOVZWL	#2160, R0		0970
					04	000C0		RET			
				7E	7C	000C1	9\$:	CLRQ	-(SP)		0979
				7E	D4	000C3		CLRL	-(SP)		
			1C	A5	DD	000C5		PUSHL	DUMP\$GL_CUR_BLOCK		
		7E	0200	8F	3C	000C8		MOVZWL	#512, -(SP)		
				65	DD	000CD		PUSHL	DUMP\$GL_BUFFER+4		
				7E	7C	000CF		CLRQ	-(SP)		
			20	AE	9F	000D1		PUSHAB	IOSB		
		04	43	A0	E9	000D4		BLBC	67(R0), 10\$		
				21	DD	000D8		PUSHL	#33		

			02	11	000DA	BRB	11\$	
			31	DD	000DC	10\$: PUSHL	#49	
		FO	A5	DD	000DE	11\$: PUSHL	DUMPSGL_CHANNEL	
			7E	D4	000E1	CLRL	-(SP)	
	66		0C	FB	000E3	CALLS	#12, SYSSQIOW	
	52		50	D0	000E6	MOVL	R0, STATUS	
	03		52	E9	000E9	BLBC	STATUS, 12\$	0980
	52		6E	3C	000EC	MOVZWL	IOSB, STATUS	
04	BC	0200	8F	B0	000EF	12\$: MOVW	#512, @BUFDESC	0981
04	A3		65	D0	000F5	MOVL	DUMPSGL_BUFFER+4, 4(R3)	0982
		1C	A5	D6	000F9	INCL	DUMPSGL_CUR_BLOCK	0983
00000870	7F		52	E8	000FC	13\$: BLBS	STATUS, 16\$	0985
	8F		52	D1	000FF	CMPL	STATUS, #2160	0986
000001F4	8F		76	13	00106	BEQL	16\$	
			52	D1	00108	CMPL	STATUS, #500	0987
0000005C	8F		6D	13	0010F	BEQL	16\$	
			52	D1	00111	CMPL	STATUS, #92	0988
00000878	8F		64	13	00118	BEQL	16\$	
			52	D1	0011A	CMPL	STATUS, #2168	0989
000000DC	8F		5B	13	00121	BEQL	16\$	
			52	D1	00123	CMPL	STATUS, #220	0990
			52	13	0012A	BEQL	16\$	
		FF54	52	DD	0012C	PUSHL	STATUS	0996
			C5	9F	0012E	PUSHAB	DUMPSGL_IDESC	0993
		00000000*	01	DD	00132	PUSHL	#1	
	67		8F	DD	00134	PUSHL	#<<<DUMPS_FACILITY@16>+4272>+2>	0994
			04	FB	0013A	CALLS	#4, LIB\$SIGNAL	
		FD1C	3A	11	0013D	BRB	15\$	0997
00000000G	00		C5	9F	0013F	14\$: PUSHAB	DUMPSGL_IRAB	1002
	52		01	FB	00143	CALLS	#1, SYSSGET	
			50	D0	0014A	MOVL	R0, STATUS	
04	BC	FD3E	C5	B0	0014D	MOVW	DUMPSGL_IRAB+34, @BUFDESC	1003
04	A3	FD44	C5	D0	00153	MOVL	DUMPSGL_IRAB+40, 4(R3)	1004
	22		52	E8	00159	BLBS	STATUS, 16\$	1005
0001827A	8F		52	D1	0015C	CMPL	STATUS, #98938	1008
			14	13	00163	BEQL	15\$	
	7E	FD24	C5	7D	00165	MOVQ	DUMPSGL_IRAB+8, -(SP)	1013
		FF54	C5	9F	0016A	PUSHAB	DUMPSGL_IDESC	1010
		00000000*	01	DD	0016E	PUSHL	#1	
	67		8F	DD	00170	PUSHL	#<<<DUMPS_FACILITY@16>+4272>+2>	1011
	52	0870	05	FB	00176	CALLS	#5, LIB\$SIGNAL	
			8F	3C	00179	15\$: MOVZWL	#2160, STATUS	1014
			52	D0	0017E	16\$: MOVL	STATUS, R0	1019
			04	00181	RET			1020

; Routine Size: 386 bytes. Routine Base: \$CODE\$ + 076D

```
1021 GLOBAL ROUTINE dump$write(recdesc): NOVALUE=
1022 BEGIN
1023
1024 Write a record to the output file
1025
1026 Inputs:
1027
1028 recdesc pointer to string descriptor for record
1029
1030 MAP
1031 recdesc : REF BBLOCK;
1032
1033
1034 dump$gl_orab[rab$w_rsz] = .recdesc[dsc$w_length];
1035 dump$gl_orab[rab$l_rbf] = .recdesc[dsc$a_pointer];
1036 IF NOT $PUT(rab=dump$gl_orab)
1037 THEN
1038 SIGNAL(
1039 dump$_facility*16 + shr$_writeerr + sts$_severe,
1040 1, dump$gl_odesc,
1041 .dump$gl_orab[rab$l_sts], .dump$gl_orab[rab$l_stv]);
1042 1 END;
```

				.EXTRN	SYSSPUT	
			0004 00000	.ENTRY	DUMPSWRITE, Save R2	1021
	52	00000000'	EF 9E 00002	MOVAB	DUMPSGL_ORAB+34, R2	
	50	04	AC D0 00009	MOVL	RECDESC, R0	1034
	62		60 B0 0000D	MOVW	(R0), DUMPSGL_ORAB+34	
06	A2	04	A0 D0 00010	MOVL	4(R0), DUMPSGL_ORAB+40	1035
		DE	A2 9F 00015	PUSHAB	DUMPSGL_ORAB	1036
00000000G	00		01 FB 00018	CALLS	#1, SYSSPUT	
	17		50 EB 0001F	BLBS	R0, 1\$	
	7E	E6	A2 7D 00022	MOVQ	DUMPSGL_ORAB+8, -(SP)	1041
		01DA	C2 9F 00026	PUSHAB	DUMPSGL_ODESC	1038
			01 DD 0002A	PUSHL	#1	
00000000G	00	00000000*	8F DD 0002C	PUSHL	#<<<DUMPS_FACILITY@16>+4304>+4>	1039
			05 FB 00032	CALLS	#5, LIB\$SIGNAL	
			04 00039 1\$:	RET		1042

; Routine Size: 58 bytes, Routine Base: \$CODE\$ + 08EF


```
939 1043 1 ROUTINE dump$close_input(fab): NOVALUE=
940 1044 BEGIN
941 1045
942 1046     Close the input file
943 1047
944 1048     Inputs:
945 1049
946 1050         fab      Address of fab
947 1051
948 1052 MAP
949 1053     fab : REF BBLOCK;
950 1054 LOCAL
951 1055     status;
952 1056
953 1057
954 1058 IF .dump$gl_flags[dump$w_records]          ! If RMS dump
955 1059 THEN
956 1060     BEGIN
957 1061         IF NOT $CLOSE(fab=.fab)          ! Close fab
958 1062     THEN
959 1063         SIGNAL(
960 1064             dump$facility*16 + shr$_closein + sts$k_error,
961 1065             1, dump$gl_idesc,
962 1066             .fab[fab$l_sts], .fab[fab$l_stv]);
963 1067     END
964 1068 ELSE
965 1069     BEGIN
966 1070         status = $DASSGN(CHAN=.dump$gl_channel);          ! Else deassign channel
967 1071         IF NOT .status
968 1072     THEN
969 1073         SIGNAL(
970 1074             dump$facility*16 + shr$_closein + sts$k_error,
971 1075             1, dump$gl_idesc,
972 1076             .status);
973 1077     END;
974 1078
975 1079     ! Free input buffer.
976 1080
977 1081
978 1082 IF .dump$gl_buffer[dsc$a_pointer] NEQ 0
979 1083 THEN
980 1084     lib$free_vm(dump$gl_buffer[dsc$w_length], dump$gl_buffer[dsc$a_pointer]);
981 1085 1 END;
```

.EXTRN SYS\$CLOSE, SYS\$DASSGN

001C 00000 DUMP\$CLOSE INPUT:

		54	00000000G	00	9E	00002	WORD	Save R2,R3,R4	1043
		53	00000000G	EF	9E	00009	MOVAB	LIB\$SIGNAL, R4	
23	00B1	C3		05	E1	00010	MOVAB	DUMP\$GL_IDESC, R3	
		52	04	AC	D0	00016	BBC	#5, DUMP\$GL_FLAGS+1, 1\$	1058
				52	DD	0001A	MOVL	FAB, R2	1061
	00000000G	00		01	FB	0001C	PUSHL	R2	
		30		50	EB	00023	CALLS	#1, SYS\$CLOSE	
							BLBS	R0, 2\$	

DUMPSMAIN
V04-000

C 13

16-Sep-1984 01:26:41

14-Sep-1984 12:21:35

VAX-11 Bliss-32 V4.0-742

DISK\$VMSMASTER:[DUMP.SRC]DUMP.B32;1

Page 40

(12)

7E	08	A2	7D	00026	MOVQ	8(R2), -(SP)	:	1066
		53	DD	0002A	PUSHL	R3	:	1063
		01	DD	0002C	PUSHL	#1	:	
	00000000*	8F	DD	0002E	PUSHL	#<<<DUMP\$ FACILITY@16>+4176>+2>	:	1064
64		05	FB	00034	CALLS	#5, LIB\$SIGNAL	:	
		1D	11	00037	BRB	2\$:	1058
	009C	C3	DD	00039	PUSHL	DUMP\$GL_CHANNEL	:	1070
00000000G	00	01	FB	0003D	CALLS	#1, SYSSDASSGN	:	
	0F	50	E8	00044	BLBS	STATUS, 2\$:	1071
		50	DD	00047	PUSHL	STATUS	:	1076
		53	DD	00049	PUSHL	R3	:	1073
		01	DD	0004B	PUSHL	#1	:	
	00000000*	8F	DD	0004D	PUSHL	#<<<DUMP\$ FACILITY@16>+4176>+2>	:	1074
64		04	FB	00053	CALLS	#4, LIB\$SIGNAL	:	
	00AC	C3	D5	00056	TSTL	DUMP\$GL_BUFFER+4	:	1082
		0F	13	0005A	BEQL	3\$:	
	00AC	C3	9F	0005C	PUSHAB	DUMP\$GL_BUFFER+4	:	1084
	00A8	C3	9F	00060	PUSHAB	DUMP\$GL_BUFFER	:	
00000000G	00	02	FB	00064	CALLS	#2, LIB\$FREE_VM	:	
		04	0006B	3\$:	RET		:	1085

; Routine Size: 108 bytes, Routine Base: \$CODE\$ + 0929

```

: 983      1086 1 ROUTINE dump$close_output: NOVALUE=
: 984      1087 BEGIN
: 985      1088
: 986      1089     Close the output file
: 987      1090
: 988      1091     Inputs:
: 989      1092
: 990      1093
: 991      1094 IF NOT $CLOSE(fab=dump$gl_ofab)
: 992      1095 THEN
: 993      1096     SIGNAL(
: 994      1097         dump$_facility^16 + shr$_closeout + sts$_error,
: 995      1098         1, dump$_gl_odesc,
: 996      1099         .dump$_gl_ofab[fab$_l_sts], .dump$_gl_ofab[fab$_l_stv]);
: 997      1100 1 END;
```

```

                                0004 00000 DUMP$CLOSE OUTPUT:
                                .WORD    Save R2
                                52 00000000' EF 9E 00002    MOVAB    DUMP$GL_OFAB, R2      : 1086
                                52 DD 00009    PUSHL    R2      : 1094
00000000G 00 01 FB 0000B    CALLS    #1, SYS$CLOSE
                                17 50 E8 00012    BLBS     R0, 1$
                                7E 08 A2 7D 00015    MOVQ     DUMP$GL_OFAB+8, -(SP)      : 1099
                                01B8 C2 9F 00019    PUSHAB   DUMP$GL_ODESC      : 1096
                                01 DD 0001D    PUSHL     #1
                                00000000* 8F DD 0001F    PUSHL     #<<<DUMP$_FACILITY@16>+4184>+2> : 1097
00000000G 00 05 FB 00025    CALLS    #5, LIB$SIGNAL
                                04 0002C 1$:    RET      : 1100
```

; Routine Size: 45 bytes, Routine Base: \$CODE\$ + 0995

```

999 1101 1 ROUTINE dump$list_width(fab): NOVALUE=
1000 1102 1 BEGIN
1001 1103 1
1002 1104 1 Determine the width of the listing line
1003 1105 1 FAB is the fab of the open file, width returned in dump$gl_width,
1004 1106 1 and dump$gl_outdesc set up as string descriptor for output buffer
1005 1107 1
1006 1108 1 MAP
1007 1109 1     fab : REF BBLOCK;
1008 1110 1 BIND
1009 1111 1     nam = .fab[fab$l_nam] : BBLOCK;
1010 1112 1 LOCAL
1011 1113 1     devnamdesc : BBLOCK[dsc$sc_s_bln],
1012 1114 1     devnambuf : VECTOR[nam$sc_maxrss, BYTE],
1013 1115 1     devnambufdesc : BBLOCK[dsc$sc_s_bln],
1014 1116 1     devinfobuf : BBLOCK[dib$sk_length],
1015 1117 1     devinfodesc : BBLOCK[dsc$sc_s_bln];
1016 1118 1 LITERAL
1017 1119 1     ch_escape = %0'033';
1018 1120 1
1019 1121 1
1020 1122 1 dump$gl_width = dump$sc_deflisiz;
1021 1123 1
1022 1124 1 devnamdesc[dsc$a_pointer] = .nam[nam$l_dev];
1023 1125 1 devnamdesc[dsc$w_length] =
1024 1126 1     CH$FIND_CH(.nam[nam$b_dev], .nam[nam$l_dev], %C':')
1025 1127 1     - .nam[nam$l_dev];
1026 1128 1 devnambufdesc[dsc$w_length] = nam$sc_maxrss;
1027 1129 1 devnambufdesc[dsc$a_pointer] = devnambuf;
1028 1130 1 $TRNLOG(LOGNAM=devnamdesc, RSLLEN=devnambufdesc, RSLBUF=devnambufdesc);
1029 1131 1 IF .devnambuf[0] EQL ch_escape
1030 1132 1 THEN
1031 1133 1     BEGIN
1032 1134 1         devnambufdesc[dsc$w_length] = .devnambufdesc[dsc$w_length] - 4;
1033 1135 1         devnambufdesc[dsc$a_pointer] = .devnambufdesc[dsc$a_pointer] + 4;
1034 1136 1     END;
1035 1137 1
1036 1138 1
1037 1139 1 ! Do a $GETDEV to get the width.
1038 1140 1
1039 1141 1 devinfodesc[dsc$w_length] = dib$sk_length;
1040 1142 1 devinfodesc[dsc$a_pointer] = devinfobuf;
1041 1143 1 IF $GETDEV(DEVNAM=devnambufdesc, SCDBUF=devinfodesc)
1042 1144 1 THEN
1043 1145 1     dump$gl_width = MINU(.devinfobuf[dib$w_devbufsiz], dump$sc_maxlisiz);
1044 1146 1
1045 1147 1
1046 1148 1 ! Set up output buffer descriptor.
1047 1149 1
1048 1150 1 dump$gl_outdesc[dsc$w_length] = .dump$gl_width;
1049 1151 1 dump$gl_outdesc[dsc$a_pointer] = dump$ab_outbuf;
1050 1152 1 END;

```

.EXTRN SYS\$TRNLOG, SYS\$GETDEV


```
000C 00000 DUMPSLIST WIDTH:
      53 00000000' EF 9E 00002 .WORD Save R2,R3      1101
      5E FE74 CE 9E 00009 MOVAB DUMPSGL_WIDTH, R3
      50 04 AC D0 0000E MOVAB -396(SPT), SP
      52 28 A0 D0 00012 MOVL FAB, R0      1111
      63 50 8F 9A 00016 MOVZBL #80, DUMPSGL_WIDTH
      FC AD 44 A2 D0 0001A MOVL 68(R2), DEVNAMDESC+4      1122
      50 39 A2 9A 0001F MOVZBL 57(R2), R0      1124
      50 3A 3A 00023 LOCC #58, R0, @68(R2)      1126
      02 12 00028 BNEQ 1$
      51 D4 0002A CLRL R1
      FB AD A2 A3 0002C 1$: SUBW3 68(R2), R1, DEVNAMDESC      1127
      7C AE FF 8F 9B 00032 MOVZBW #255, DEVNAMBUFDESC      1128
      0080 CE 0084 CE 9E 00037 MOVAB DEVNAMBUF, DEVNAMBUFDESC+4      1129
      7E 7C 0003E CLRQ -(SP)      1130
      7E D4 00040 CLRL -(SP)
      0088 CE 9F 00042 PUSHAB DEVNAMBUFDESC
      008C CE 9F 00046 PUSHAB DEVNAMBUFDESC
      F8 AD 9F 0004A PUSHAB DEVNAMDESC
      00000000G 00 06 FB 0004D CALLS #6, SYS$TRNLOG
      1B 0084 CE 91 00054 CMPB DEVNAMBUF, #27      1131
      7C AE 04 A2 00058 BNEQ 2$
      0080 CE 04 C0 0005F SUBW2 #4, DEVNAMBUFDESC      1134
      04 AE 74 8F 9B 00064 ADDL2 #4, DEVNAMBUFDESC+4      1135
      08 08 AE 9E 00068 MOVZBW #116, DEVINFODESC      1141
      SE DD 0006D MOVAB DEVINFOBUF, DEVINFODESC+4      1142
      7E 7C 0006F PUSHL SP      1143
      7E D4 00071 CLRQ -(SP)
      008C CE 9F 00073 CLRL -(SP)
      05 FB 00077 PUSHAB DEVNAMBUFDESC
      50 E9 0007E CALLS #5, SYS$GETDEV
      0E AE 3C 00081 BLBC R0, 4$
      0084 8F 50 B1 00085 MOVZWL DEVINFOBUF+6, R0      1145
      04 1B 0008A CMPW R0, #132
      50 8F 9A 0008C BLEQU 3$
      63 50 D0 00090 MOVZBL #132, R0
      F4 A3 63 B0 00093 3$: MOVL R0, DUMPSGL_WIDTH
      F8 A3 FF70 C3 9E 00097 4$: MOVW DUMPSGL_WIDTH, DUMPSGL_OUTDESC
      RET MOVAB DUMPSAB_OUTBUF, DUMPSGL_OUTDESC+4      1150
      04 0009D      1151
      04 0009D      1152
```

; Routine Size: 158 bytes, Routine Base: \$CODE\$ + 09C2

```
1052 1153 1 ROUTINE dump$file_error(message,fab,sts,stv): NOVALUE=
1053 1154 2 BEGIN
1054 1155 3 ++
1055 1156 4 FUNCTIONAL DESCRIPTION:
1056 1157 5
1057 1158 6 This routine signals an error for a file.
1058 1159 7
1059 1160 8 Inputs:
1060 1161 9
1061 1162 10 message      Message
1062 1163 11 fab          Address of the fab
1063 1164 12 sts, stv     STS and STV values
1064 1165 13
1065 1166 14 --
1066 1167 15
1067 1168 16 MAP
1068 1169 17 fab : REF BBLOCK;
1069 1170 18 BIND
1070 1171 19 nam = .fab[fab$_l_nam] : BBLOCK;
1071 1172 20 LOCAL
1072 1173 21 filedesc : BBLOCK[dsc$_s_bln];
1073 1174 22
1074 1175 23 CH$FILL(0, dsc$_s_bln, filedesc);
1075 1176 24
1076 1177 25 IF .nam[nam$_rsl] NEQ 0 ! If resultant name present
1077 1178 26 THEN
1078 1179 27 BEGIN
1079 1180 28 filedesc[dsc$_w_length] = .nam[nam$_rsl];
1080 1181 29 filedesc[dsc$_a_pointer] = .nam[nam$_rsa];
1081 1182 30 END
1082 1183 31 ELSE IF .nam[nam$_esl] NEQ 0 ! If expanded name present
1083 1184 32 THEN
1084 1185 33 BEGIN
1085 1186 34 filedesc[dsc$_w_length] = .nam[nam$_esl];
1086 1187 35 filedesc[dsc$_a_pointer] = .nam[nam$_esa];
1087 1188 36 END
1088 1189 37 ELSE
1089 1190 38 BEGIN
1090 1191 39 filedesc[dsc$_w_length] = .fab[fab$_fns]; ! Use filename string
1091 1192 40 filedesc[dsc$_a_pointer] = .fab[fab$_fna]; ! if all else fails
1092 1193 41 END;
1093 1194 42
1094 1195 43
1095 1196 44
1096 1197 45 SIGNAL(.message, 1, filedesc, .sts, .stv);
1097 1198 46 END; ! Of dump$file_error
```

00FC 0000 DUMPSFILE ERROR:

				08	C2	00002	WORD	Save R2,R3,R4,R5,R6,R7	: 1153
		5E					SUBL2	#8, SP	
		57	08	AC	D0	00005	MOVL	FAB, R7	: 1171
		56	28	A7	D0	00009	MOVL	40(R7), R6	
08	00	6E		00	2C	0000D	MOVCS	#0, (SP), #0, #8, FILEDESC	: 1176

H 13
16-Sep-1984 01:26:41
14-Sep-1984 12:21:35

Page 45
(15)

		03	6E A6 0B	95 00013 13	00012 00013 00016	TSTB BEQL	3(R6) 1\$: :	1178
	6E	03	A6	9B	00018	MOVZBW	3(R6), FILEDESC	:	1181
04	AE	04	A6	D0	0001C	MOVL	4(R6), FILEDESC+4	:	1182
			19	11	00021	BRB	3\$:	1178
		0B	A6	95	00023	TSTB	11(R6)	:	1184
			0B	13	00026	BEQL	2\$:	
	6E	0B	A6	9B	00028	MOVZBW	11(R6), FILEDESC	:	1187
04	AE	0C	A6	D0	0002C	MOVL	12(R6), FILEDESC+4	:	1188
			09	11	00031	BRB	3\$:	
	6E	34	A7	9B	00033	MOVZBW	52(R7), FILEDESC	:	1192
04	AE	2C	A7	D0	00037	MOVL	44(R7), FILEDESC+4	:	1193
	7E	0C	AC	7D	0003C	MOVQ	STS, -(SP)	:	1197
		0B	AE	9F	00040	PUSHAB	FILEDESC	:	
			01	DD	00043	PUSHL	#1	:	
		04	AC	DD	00045	PUSHL	MESSAGE	:	
00000000G	00		05	FB	00048	CALLS	#5, LIB\$SIGNAL	:	
			04	0004F	RET			:	1198

; Routine Size: 80 bytes, Routine Base: \$CODES + 0A60

: 1099 1199 1 END
: 1100 1200 0 ELUDOM

.EXTRN LIB\$SIGNAL, LIB\$STOP

PSECT SUMMARY

Name	Bytes	Attributes
\$GLOBALS	804	NOVEC, WRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
\$OWNS	40	NOVEC, WRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
-LIB\$KEYOS	8	NOVEC, NOWRT, RD, EXE, SHR, LCL, REL, CON, PIC, ALIGN(1)
-LIB\$STATES	86	NOVEC, NOWRT, RD, EXE, SHR, LCL, REL, CON, PIC, ALIGN(1)
-LIB\$KEY1\$	17	NOVEC, NOWRT, RD, EXE, SHR, LCL, REL, CON, PIC, ALIGN(1)
\$CODES	2736	NOVEC, NOWRT, RD, EXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
\$SPLITS	358	NOVEC, NOWRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)

Library Statistics

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
\$255\$DUA28:[SYSLIB]STARLET.L32;1	9776	178	1	581	00:00.8
- \$255\$DUA28:[SYSLIB]TPAMAC.L32;1	42	28	66	14	00:00.1

COMMAND QUALIFIERS

: BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS\$:DUMP/OBJ=OBJ\$:DUMP MSRC\$:DUMP/UPDATE=(ENH\$:DUMP)

: Size: 2736 code + 1313 data bytes
: Run Time: 00:36.6
: Elapsed Time: 02:26.3
: Lines/CPU Min: 1965
: Lexemes/CPU-Min: 55923
: Memory Used: 319 pages
: Compilation Complete

0123

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY